SEGUIMIENTO A LAS BARRAS BRAVAS DEL FUTBOL EN LA CIUDAD DE PEREIRA, BASADO EN TECNOLOGÍAS BIG DATA

CÉSAR ADOLFO GONZÁLEZ MARÍN
Diciembre 2017

Proyecto de Grado para optar al título de
MAGISTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Director
PhD. JULIO CÉSAR CHAVARRO PORRAS

UNIVERSIDAD TECNOLOGICA DE PEREIRA
FACULTAD DE INGENIERIAS: ELECTRICA, ELECTRONICA, FISICA Y CIENCIAS DE LA COMPUTACION
MAESTRIA EN INGENIERIA DE SISTEMAS Y COMPUTACION

Abstract

The project seeks to find a model for the follow-up of fans of the roving bars of the Deportivo Pereira soccer team, based on Big Data technologies. The social network Twitter is taken to make the observations, and the programming language R with the RStudio interface to carry out the implementation, in which by means of filters and lexicons a set of messages known as knowledge is delivered as a result. The standard process model for data mining CRISP - DM (CRISP - DM: Towards a Standard Process Model for Data Mining) is taken as the base methodology.

Keywords: CRISP-DM, Microblogging, Big Data, Data mining, R language, Twitter.

Tabla de Contenidos

Lista de Tablas vi
Lista de Figuras vii
Capítulo 1 Introducción 1
1.1 Definición del problema 1
1.2. Justificación 3
1.3 Objetivo general 5
1.4 Objetivos específicos 5
1.4 Diseño metodológico 6
Capítulo 2 Marco de Referencia 9
2.1. Definición de Big Data 9
2.2. Tipos de datos 11
2.2.1. Datos estructurados 11
2.2.2. Datos semiestructurados 11
2.2.3. Datos no estructurados 11
2.3. Características de Big Data 12
2.3.1. Volumen 12
2.3.2. Velocidad 13
2.3.3. Variedad 13
2.3.4. Veracidad 13
2.3.5. Valor 13
2.4. Big Data Analytics 14
2.4.1. Adquisición de datos 15
2.4.2. Organización de los datos 15
2.4.3. Análisis de información 15
2.4.4. Decisión 15
2.5. Áreas de análisis de datos 16
2.5.1. Analítica de datos (analytics) 17
2.5.2. Analítica WEB 17
2.5.3. Analítica social 17

2.5.4. Analítica móvil 17
2.5.5. Analítica de Big Data 18
2.5.6. Analítica de M2M (machine to machine) 18
2.6. Analítica social 19
2.6.1. Social Media 19
2.6.2. Herramientas de analítica social 19
2.6.3. Análisis de sentimientos 20
2.6.4. Herramientas de investigación y monitorización en social media 22
2.7. Arquitectura de Big Data 27
2.8. Herramientas teóricas que dan soporte a las tecnologías Big Data 28
2.8.1. Google File System (GFS) 29
2.8.2. MapReduce 47
2.8.3. Bigtable 53
Capítulo 3 Técnicas y Tecnologías 64
3.1 Minería de datos 64
3.2 Minería de opinión 66
3.3 Microblogging 68
3.4 Twitter 68
3.5 Análisis de sentimientos en twitter 70
3.6 Facebook 72
3.7 R 75
Capítulo 4 Implementación 77
4.1 Diagrama de bloques para el desarrollo del proyecto 77
4.1.1 Recolección de datos 79
4.1.2 Almacenamiento 79
4.1.3 Técnicas de filtrado 80
4.1.4 Análisis 81
4.1.5 Conocimiento 81
4.2 Ejecución del proyecto 81
4.2.1 Recolección de datos 82

4.2.2 Almacenamiento 83
4.2.3 Técnicas de filtrado 91
4.2.4 Análisis 92
4.2.5 Conocimiento 93
4.3 Códigos en R 93
4.3.1 Presentación del software empleado 94
4.3.2 Instalación y carga de librerías 94
4.3.3 Autenticación a Twitter 96
4.3.4 Almacenamiento 96
4.3.5 Técnicas de filtrado 98
4.3.6 Análisis con R 98
4.3.7 Conocimiento 100
Capítulo 5 Análisis de Resultados y Conclusiones 102
5.1 Recopilación y lectura de la información 102
5.2 Conclusiones 111
Capítulo 6 Lista de Referencias 115
6.1 Referencias 115

Lista de Tablas

pág

Tabla1: Herramientas para investigación y monitorización en social media.…..…. 27

Tabla 2: Descripción de las acciones de ejecución de MapReduce.…… 51

Tabla 3. Listado de categorías de búsqueda ……..…….. 102

Tabla 4. Muestra de mensajes obtenidos por categoría……..…….. 104

Tabla 5. Cantidad de mensajes repetidos dentro del Data Set …….…… 105

Tabla 6. Listado de palabras con más frecuencia dentro de data set filtrado…….. 107

Tabla 7. Listado de las 100 palabras con más frecuencia dentro de Data Set.……. 108

Tabla 8. Listado de palabras con frecuencia y marca por contenido ofensivo…..... 109

Tabla 9. Listado de lugares en que ocurren confrontaciones …….. 113

Lista de Figuras

pág

Figura 1: Fases modelo de proceso estándar para minería de datos CRISP – DM.. 6

Figura 2: Tipos de datos Big Data ……..…….. 11

Figura 3: Las 5V en Big Data. …….…… 12

Figura 4: Etapas en el tratamiento de los grandes volúmenes de datos.…….…. 17

Figura 5: Propuesta a la relación entre las categorías de analítica del análisis de datos en Big Data. …….…….. 18

Figura 6: Arquitectura de referencia de Big Data propuesta por Sunil Soares.…... 27

Figura 7: Arquitectura de referencia de Big Data. Fuente Soares. Adaptada.…… 28

Figura 8: Arquitectura de Google File System (GFS)....…….……...…….. 32

Figura 9. Información general de ejecución de MapReduce.……..……. 50

Figura 10. Componentes principales de la implementación de Bigtable.……..….. 58

Figura 11. Jerarquía y composición de una tabla en Bigtable.……..…….. 59

Figura 12. Representación de una tabla en GFS.……..……... 62

Figura 13. Proceso KDD.……..... 65

Figura 14. Contenido de los tweets ……. 72

Figura 15. Usuarios de Facebook entre 2004 y 2015……. 74

Figura 16. Porcentaje de usuario en Facebook por edad.….…… 74

Figura 17. Países hispanos con más usuarios en Facebook.…… 75

Figura 18. Logo. Fuente r-project.org……..……..…. 76

Figura 19. Modelo de bloques del proceso en el proyecto ……..…… 77

Figura 20. Desarrollo del modelo de bloques del proceso en el proyecto.……... 78

Figura 21. Etapa de almacenamiento.……...……... 83

Figura 22. Dominio y rango en el proceso de extracción de datos…….……...….. 84

Figura 23. Estrategias de extracción de información con la API TwitteR…….. 85

Figura 24. Procedimiento de filtrado de mensajes …….. 92

Capítulo 1
Introducción

1.1 Definición del problema

Desde hace por lo menos una década, los medios noticiosos nacionales han informado sobre las agresiones físicas que se ocasionan los jóvenes hinchas de diferentes equipos a lo largo de nuestra geografía - *como lo muestra el periodista colombiano Guillermo Arturo Prieto La Rotta en el programa Especiales Pirry del canal de televisión RCN, capitulo emitido el 22 de junio de 2008 llamado"Barras bravas en Colombia: 90 minutos de una tragedia"*, Pereira no ha sido ajena a esta problemática y es así como en sitios públicos de nuestra ciudad como el Estadio Olímpico Hernán Ramírez Villegas, sector del Viaducto Cesar Gaviria Trujillo, la Plaza Cívica Ciudad Victoria e instalaciones de Mega bus [1], entre otros, se han presentado encuentros violentos entre barras bravas; el periódico El Tiempo publicó en uno de sus artículos el 30 de junio de 2013: "Cada día asesinan a un hincha del futbol en Colombia", y las agresiones han continuado. El científico Colombiano Raúl Cuero Rengifo escribe en su último libro "La Orfandad de la nueva generación" *El sentimiento de orfandad es quizás el común denominador de todos los seres humanos.*…y allí mismo más adelante… *enfrentamos este sentimiento mediante diversas actividades.* En su búsqueda del ser y a falta de proyecciones que les permitan desarrollar su creatividad, inventiva e innovación, jóvenes encuentran refugio en el futbol al apoyar a sus equipos de preferencia, y en los grandes avances de la tecnología de la comunicación como una forma de sentirse asociado con sus congéneres (Cuero, 2013) [2].

Para el Alcalde de Pereira Enrique Vásquez Zuleta las barras bravas han causado importantes dificultades en la ciudad de Pereira [3], en su momento para el subcomandante de la Policía Metropolitana Coronel Delbert Plata San Jorge a su llegada a la ciudad manifestó que sería su

prioridad controlar las barras bravas, “Yo sé y tengo conocimiento de como son los encuentros deportivos y como los jóvenes se salen de control y protagonizan actos violentos, esa es una de las dificultades que he identificado en Pereira y hay que seguir combatiendo, así como la disminución del homicidio y los enfrentamientos por micro tráfico de estupefacientes” [4]). Las riñas y agresiones pasan por contravenciones hasta delitos de mayor impacto como hurto, daño en bien ajeno y hasta homicidio (como lo informan los medios de comunicación), la investigación, recurso electrónico “Barras bravas en el futbol: consumo de drogas y violencia” de la Fundación Universitaria Luis Amigó [5] brinda un panorama actualizado de la situación en la ciudad de Medellín con sus dos equipos más representativos.

Por su parte las redes sociales son un vehículo más que permite realizar provocaciones abiertas o concretas, pactar citas para reñir, como al parecer ocurrió en el asesinato del joven de 19 años Andrés Camilo Valverde Armenta en marzo de 2015, según sus familiares las barras se citaron por Facebook para pelear [6]; en la localidad de Bosa en Bogotá, habitantes en diálogos con la cadena de noticias Caracol en diciembre 2013, informaron de los constantes encuentros entre hinchas de dos reconocidos equipos del Futbol Colombiano para reñir en sus calles, lo que ha dejado heridos, dicen que estas citas se hacen por medio redes sociales [7]; en julio de 2012 las barras organizadas del Deportivo Pereira y Atlético Nacional protagonizaron disturbios antes y después del encuentro futbolero, violencia pactada al parecer por redes sociales [8].Pero, además de redes sociales son muchas las herramientas tecnológicas por las cuales se pueden llevar a cabo estas prácticas como por ejemplo blogs, mensajes de texto, llamas por celular, mensajes de correo electrónico, WhatsApp. Lo anterior propiciado por los esfuerzos en cobertura de los servicios de banda ancha como política de gobierno y al crecimiento en el uso del smartphone con planes de datos 3G y 4G en todos los estratos.

Las redes sociales hacen parte de los grandes volúmenes de datos que las tecnologías de la información generan. En el año 2000, se almacenaron en el mundo 800.000 petabytes. Se espera que para el año 2020 se alcancen 35 zettabytes. Solo Twitter genera más de 9 terabytes (TB) de datos cada día. Facebook 10 TB (Joyanes, 2013) [9], esta información adquiere connotación de información masiva y por ello mismo se hace difícil de procesar, almacenar y entender desde un punto de vista computacional debido a la variedad de formas que expresa ya que no son solo datos estructurados para las bases de datos relacionales (datos tradicionales) sino datos que no tienen formatos fijos o no tienen estructura uniforme, son los denominados datos semiestructurados y no estructurados respectivamente (Joyanes, 2013) [9].

Por su parte las tecnologías Big Data están diseñadas para capturar y analizar grandes volúmenes de datos para extraer valor de ellos en un tiempo razonable en el que la información sea oportuna. Es ahí donde estas tecnologías podrían ofrecer una alternativa de seguimiento si lograran entregar a entidades de gobierno información de valor para tomar decisiones en la mitigación de este fenómeno social, en advertir confrontaciones entre grupos de barras bravas y en esclarecer delitos conexos con este tipo de comportamientos.

¿Es posible desarrollar un modelo de seguimiento a las barras bravas del futbol basado en tecnologías Big Data, para la ciudad de Pereira?

1.2. Justificación

La violencia entre jóvenes integrantes de las barras bravas de los equipos de futbol colombiano se manifiesta también en la ciudad de Pereira, daños en bien público y privado, lesiones personales, hurtos, consumo de alcohol y sustancias psicoactivas y en algunos casos homicidios son comportamientos asociados a estos grupos. Como lo muestra el estudio del grupo

de investigación *Farmacodependencia y otras adicciones* de la fundación universitaria Luis Amigó denominado *"Barras bravas en el futbol: consumo de drogas y violencia"* en la ciudad de Medellín en 2014. Entre otros hallazgos importantes en este estudio, el 43% de los sujetos incluidos en él, creen que el consumo de drogas es una de las causas de la violencia en los estadios y alrededores, el 40% de las personas en el estudio que dicen ser hinchas de algún equipo (Nacional o Medellín en este caso) han participado en actos violentos durante el partido de futbol y lo han hecho bajo el efecto del alcohol y las drogas, las posibilidades de consumo de sustancias permitidas y prohibidas antes y después de un encuentro de futbol son altas, las personas relacionadas con barras bravas y que participan de actos violentos en general son hombres con nivel de escolaridad secundaria, estudiantes, entre 13 y 24 años de edad, solteros y consumidores de sustancias, la asociación más significativa entre consumo de sustancias sicoactivas y actos violentos se encontró con la marihuana, cocaína, Sacol (pagamento) y Rivotril.

Por otro lado, se ha presumido y comprobado que los encuentros de grupos de barras bravas se promocionan por medio de las redes sociales, en sitios distantes de los estadios de futbol como al parecer ocurrió en el asesinato del joven de 19 años Andrés Camilo Valverde Armenta en marzo de 2015, según sus familiares las barras se citaron por Facebook para pelear, o como en la localidad de Bosa de Bogotá, habitantes informaron en 2013 de los constantes encuentros entre hinchas de dos recocidos equipos de futbol para reñir en sus calles, lo que ha dejado heridos, dicen que estas citas se hacen por medio redes sociales.

La cantidad de información generada por las personas con sus equipos tecnológicos y por sensores, por medio de redes sociales, es considerada como *información no estructurada o datos*

Big Data. Esta clasificación viene de dos grandes grupos, los datos estructurados o datos tradicionales y los datos no estructurados, clasificación que al mismo tiempo puede incluir *datos semi estructurados*. El uso cada vez mayor de terminales móviles tipo smartphone conectados a la web es una oportunidad porque permite realizar seguimiento a las manifestaciones de las barras bravas de futbol en la ciudad de Pereira, por medio de un modelo basado en tecnologías Big Data.

1.3 Objetivo general

Proponer un modelo de seguimiento y verificación al comportamiento de las barras bravas de futbol en Pereira, soportado en tecnologías Big Data y restringido a la red social Twitter.

1.4 Objetivos específicos

1. Estudiar y estructurar algunos de los métodos y tecnologías utilizados en Big Data para el tratamiento de datos no estructurados y que puedan ser aplicables a la información obtenida de redes sociales.

2. Determinar, seleccionar y poblar un modelo de recolección de información de la red social Twitter, en torno a eventos específicos y restringidos a una locación geográfica determinada.

3. Evaluar y seleccionar las técnicas de filtrado y de análisis de datos, aplicables a la información proveniente de la red social Twitter.

4. Filtrar y analizar la información obtenida de la red social Twitter, aplicando las técnicas seleccionadas de Big Data, para predecir encuentros de barras bravas en la ciudad de Pereira.

1.4 Diseño metodológico

Como metodología del presente proyecto, se toma la definición del modelo de proceso estándar para minería de datos CRISP – DM (CRISP-DM: Towards a Standard Process Model for Data Mining) [10], en las fases y tareas respectivas del mismo así como para sus salidas.

Las fases del modelo de proceso estándar para minería de datos CRISP – DM se muestran en la figura 1.

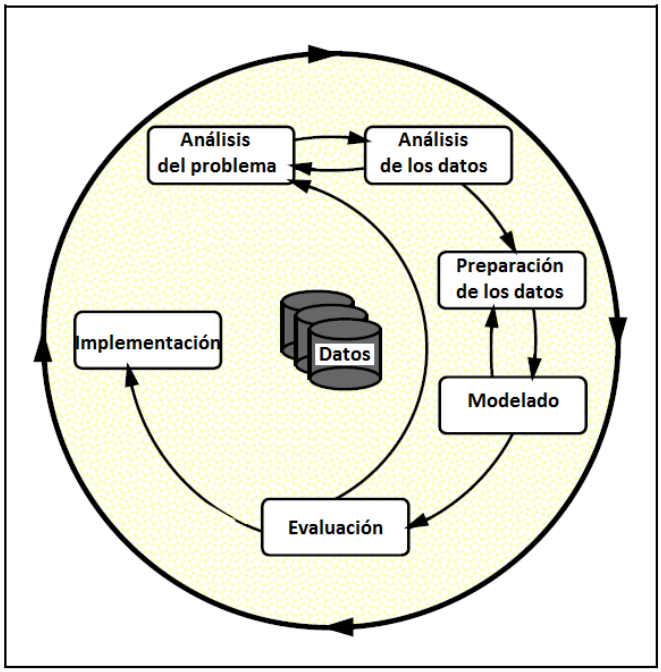


Figura 1. Fases modelo de proceso estándar para minería de datos CRISP – DM.

A continuación, la descripción de las fases de la aplicación del modelo CRISP – DM en la ejecución del presente proyecto.

Análisis del problema: Una de las maneras en que personas de la ciudad de Pereira usan su tiempo libre es a través de las barras de futbol. Pertenecer a alguna de ellas es una alternativa recreativa, pero no en todos los casos termina siendo así. Investigaciones realizadas y

consignadas en la definición del problema del presente documento así lo muestran. La violencia ha rodeado el futbol desde sus orígenes y de manera particular después de la segunda guerra mundial, siendo más fuerte a partir de la década de los años ochenta, el futbol se ha estado en medio de la violencia y muerte, como se observa en el resumen cronológico de antecedentes violentos en los estadios (Cañon y García, 2007) [11], sumado al hecho que el uso de las tecnologías móviles esta en creciente desarrollo. Dificultades como la expuesta y que nuestra sociedad ha venido padeciendo podría mitigarse, entre otras maneras, utilizando tecnologías Big Data, así *¿Es posible desarrollar un modelo de seguimiento a las barras bravas del futbol basado en tecnologías Big Data, para la ciudad de Pereira?*

Análisis de los datos: Durante esta etapa se hace necesario el entendimiento de los datos contenidos en los Data Set producidos por la red social Twitter, reconocer la estructura de sus mensajes y contemplarlos frente a los objetivos del proyecto.

Preparación de los datos: Conocidas las características del mensaje, se toman sub conjuntos de los datos a partir de las Data Set por medio de la aplicación de filtros en características de edad, ubicación geográfica de emisión y de recepción del mensaje; así como dentro del cuerpo mismo del mensaje como por ejemplo, palabras que hagan referencia a nombres de equipos de futbol colombiano, apodos o sitios representativos de la ciudad.

Modelado: En esta fase, se realizan los modelos con los que se realizará el seguimiento a las barras bravas de la ciudad de Pereira.

Evaluación: En esta fase del proyecto se consideran los modelos realizados, siendo quizás necesario redefinir el modelo o regresar a alguno de los pasos anteriores.

Implementación: En esta etapa del proyecto se presenta el modelo en ejecución, tal vez sea necesario aprender del ciclo actual para iniciar con uno nuevo y refinar el modelo.

Capítulo 2
Marco de Referencia

2.1. Definición de Big Data

Big Data es la agrupación de múltiples tendencias tecnológicas, maduradas a partir del año 2000. Dichas tecnologías se han consolidado entre los años 2011 a 2013, en el momento en el que la sociedad mundial se encuentra generando información alrededor de la geo - referenciación, redes sociales, banda ancha, reducción de los costes de conexión a internet, medios sociales, telefonía móvil, internet de las cosas y computación en la nube. Lo anterior tiene en común, los grandes volúmenes de datos generados, la necesidad de capturarlos, almacenarlos y analizarlos.

Para la consultora tecnológica IDC en 2012, “Big Data es una nueva generación de tecnologías, arquitecturas y estrategias diseñadas para capturar y analizar grandes volúmenes de datos provenientes de múltiples fuentes heterogéneas a una alta velocidad con el objeto de extraer valor económico de ellos” (Consultora IDC, 2012) [12].

La consultora Gartner en 2011, escribe, “Big Data son los grandes conjuntos de datos que tiene tres características principales: volumen (cantidad), velocidad (velocidad de creación y utilización) y variedad (tipos de fuentes de datos no estructurados, tales como interacción social, video, audio, cualquier cosa que se pueda clasificar en una base de datos)” (Consultora Gartner, 2011) [13].

La definición de McKinsey Global Institute en 2011, “Big Data se refiere a los conjuntos de datos cuyo tamaño está más allá de las capacidades de las herramientas típicas de software de bases de datos para capturar, almacenar, gestionar y analizar” (Consultora McKinsey, 2011) [14].

En la publicación “Toward Scalable Systems for Big Data Analytics: A Technology Tutorial” (Han, Yonggang y Tat-Seng Chua, 2014) [15], los autores referencian al Instituto Nacional de Estándares y Tecnología (NIST) para la definición de Big Data, “Big Data es donde el volumen de datos, la velocidad de adquisición, o la representación de datos limita la capacidad de realizar análisis efectivo utilizando enfoques relacionales tradicionales o requiere del uso de escalamiento horizontal significativo para el procesamiento eficiente”. En el mismo artículo se realiza una clasificación plausible a la definición de Big Data, así:

- Ciencia de Big Data.
- *Frameworks* de Big Data.

La ciencia de Big Data es “El estudio de técnicas que abarcan la adquisición, acondicionamiento y evaluación de grandes datos”, y en lo que respecta a *frameworks* de Big Data, “Librerías de software junto con sus algoritmos asociados que permiten distribuir el procesamiento y análisis a grandes datos a través de *clusters* de computadoras”.

En “Concept Definition for Big Data Architecture in the Education System” (Michalik, Stofa y Zolotova, 2014) [16], los autores utilizan la definición que IBM hace al término Big Data de la siguiente manera, “Los datos procedentes de todas partes; sensores utilizados para recopilar

información climática, mensajes de redes sociales, fotos digitales y vídeos, registros de transacciones, y señales GPS del teléfono celular, para nombrar unos pocos.”

2.2. Tipos de datos

En Big Data los datos son diferentes a los datos tradicionales es decir los datos estructurados almacenados en bases de datos relacionales. Los datos se consideran en dos tipos, los estructurados y los no estructurados (ver figura 2).

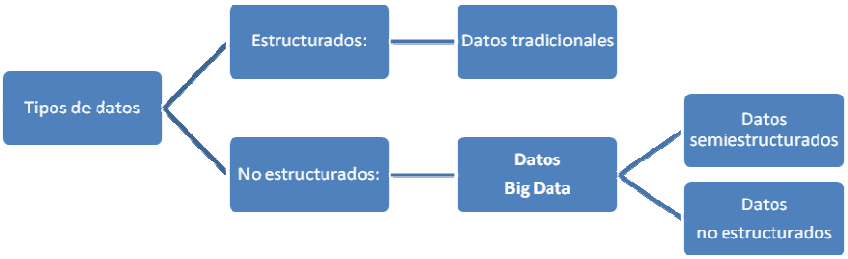


Figura 2. Tipos de datos Big Data.

2.2.1. Datos estructurados: son aquellos datos con formato y campos fijos, en el que el formato es anticipadamente definido, para ser almacenados en bases de datos relacionales; este tipo de datos guardan un orden específico lo que facilita trabajar con ellos.

2.2.2. Datos semiestructurados: son aquellos datos que no tienen formatos fijos, pero que contienen etiquetas, marcadores o separadores que permiten entenderlos; se procesan a base de reglas para extraer la información en piezas. Los lenguajes XML y HTML son ejemplos de texto con etiquetas.

2.2.3. Datos no estructurados: son aquellos datos que no tienen formatos predefinidos, es decir no tienen estructura uniforme. Se tiene poco o ningún control sobre su estructura. Por ejemplo

los correos electrónicos, mensajes instantáneos SMS, WhatsApp, Viber, fotos, audios, videos entre otros.

2.3. Características de Big Data

IBM y Gartner plantean tres dimensiones para el entendimiento de la naturaleza de los Big Data, conocido como el modelo de las 3V; inclusive IBM considera una cuarta V correspondiente a la *veracidad*, y otras fuentes añaden una más, el *valor*. Sin embargo, las distintas fuentes tiene en común el modelo de las 3V que corresponde a *volumen, velocidad* y *variedad*. La figura 3 muestra el esquema de las 5V en Big Data, ampliando al propuesto por IBM inicialmente.

Figura 3. Las 5V en Big Data.

2.3.1. Volumen: se espera que entre el presente año y el 2020, la humanidad entre en la era de zettabyte; en el año 2000 se almacenaron en el mundo 800.000 petabytes y se proyecta que para el año 2020 se alcancen 35 zettabytes (ZB); en 2012 Twitter generó más de 9 terabytes (TB) de datos al día. La gran cantidad de datos que las organizaciones generan es mayor día tras día, es por eso que hablar de un gran tamaño de datos en estos momentos es dar lugar a que solo en unos

semestres ese valor no sea tan grande para ese momento. El volumen en Big Data hace referencia al presente de la cantidad de información que genera y almacena una organización, es decir a su volumen masivo de datos. Para IBM el volumen de datos disponible que tiene una organización está en ascenso en contraste con la disminución en el análisis que se hace de ellos.

2.3.2. Velocidad: en Big Data éste tema merece consideración ya que el aumento de los flujos de datos en las organizaciones aumenta la velocidad en que se deben almacenar datos y sugiere últimas versiones de los gestores de grandes bases de datos. Este aumento en la velocidad de los datos requiere que el procesamiento de ellos se haga en tiempo real para mejorar la toma de decisiones. En Big Data velocidad hace referencia a la velocidad con la que fluyen los datos desde distintas fuentes como sensores, chips de RFID, chips NFC, datos de geolocalización, logs, entre otros.

2.3.3. Variedad: representa todos los tipos de datos, entre datos estructurados y no estructurados. Las fuentes de datos son de cualquier tipo y ello aumenta la complejidad; por ejemplo los videos no pueden ser tratados en bases de datos relacionales como si se puede hacer con el registro de ingreso de los empleados a una empresa. Considerar toda esta variedad de tipos de archivos supone un reto en la clasificación de los datos para que el análisis de ellos entregue resultados de valor a quien los investiga.

2.3.4. Veracidad: en Big Data este término se relaciona a la fiabilidad de las fuentes de datos, debido al aumento de fuentes de ellos, y además a la variedad en los tipos de datos.

2.3.5. Valor: en Big Data se hace referencia a esta característica en el sentido de la rentabilidad y eficiencia con la que se puede obtener información útil a partir de la información masiva con la que se cuenta, bien sea por la implementación de tecnologías pagadas o las tecnologías de código

abierto implementadas en la organización, así como de manera alquilada, las PaaS (plataformas como servicio) o IaaS (infraestructuras como servicio).

2.4. Big Data Analytics

La analítica de Big Data es el proceso de examinar con gran velocidad, conjuntos de grandes volúmenes de datos entre una amplia variedad de tipos y descubrir patrones ocultos, nuevas correlaciones y más información útil, en un tiempo razonable en el que la oportunidad de la información proporcione ventajas competitivas al investigador.

Los grandes volúmenes de información pueden proceder de fuentes de datos no estructurados como los que generan smartphones, medios de comunicación, información suministrada por sensores, actividades sociales, entre otros; pero, además pueden proceder de datos estructurados almacenados en bases de datos relacionales.

El análisis de grandes datos (analítica de Big Data o Big Data analytics) corresponde a datos estructurados, no estructurados y semiestructurados.

El análisis de grandes datos relacionales se puede realizar con herramientas de software tradicionales con técnicas sencillas o avanzadas como minería de datos o análisis predictivo, análisis estadísticos, programación compleja de SQL; en cuanto a las fuentes de datos no estructurados, pueden no encajar dentro de los esquemas de los almacenes de datos tradicionales o EDW (Enterprise Data Warehouse) o no estar en capacidad de atender la demanda de procesamiento de datos requerido.

Para atender la demanda de procesamiento de grandes datos han surgido tecnologías de bases de datos distintas a las relacionales llamadas bases de datos NoSQL, bases de datos en memoria y

MapReduce. Este sistema se integra a través de un cluster, y bien puede ser por medio de software de código abierto o propietario. Grandes compañías como HP, IBM, Oracle y Microsoft han desarrollado importantes productos propietarios.

Para el tratamiento de los grandes volúmenes de datos se requieren las siguientes etapas, como la se observa en la figura:

2.4.1. Adquisición de datos: los datos proceden de fuentes de datos diversas, es decir, de fuentes de datos tradicionales como almacenes de datos, bases de datos relacionales, entre otros; y, de fuentes de datos no estructurados o semi estructurados. Los datos procedentes de ambos tipos de fuentes de datos, pueden ser almacenados en bases de datos NoSQL o en bases de datos "en memoria".

2.4.2. Organización de los datos: el origen distinto en las fuentes de datos requiere que luego de que se adquiera la información, deba prepararse, siendo tal vez necesario eliminar datos o parte de ellos para dejar lo más relevante de estos.

2.4.3. Análisis de información: es una etapa muy importante dentro del tratamiento de los grandes volúmenes de datos. Consiste en analizar todos los datos por medio de herramientas estadísticas avanzadas como minería de información, minería social, herramientas desarrolladas para diseño de estadística avanzada como el lenguaje de programación R.

2.4.4. Decisión: es en esta etapa en donde con los resultados obtenidos del análisis de información se obtiene conocimiento, preferiblemente en tiempo real; para que se incluya en los tableros de control, cuadros de mando y herramientas de visualización, y así predecir el comportamiento que va a tener el objeto de estudio.

2.5. Áreas de análisis de datos

Según ISACA (Information Systems Audit and Control Association - *Asociación de Auditoría y Control de Sistemas de Información*) en 2011, la analítica de datos "implica los procesos y actividades diseñados para obtener y evaluar datos para extraer información útil".

La analítica de datos es la ciencia que examina datos en bruto o crudos para obtener conclusiones acerca de la información contenida en ellos.

Existen muchas herramientas de software diseñadas para analítica de datos (herramientas como CRM (Customer Relationship Management - Administración de la Relación con los Clientes), OLAP (On-Line Analytical Processing), tableros de control, entre otros), en las que se utilizan técnicas de consultas (quering), reportes (reporting), visualización, lógica difusa, minería de datos, análisis de datos predictivos, streaming de audio, video o fotografía, entre otros.

La analítica de datos está influenciada por todo tipo de dispositivos como GPS, sensores, dispositivos tipo internet de las cosas, chips NFC y RFID, códigos de barras, códigos QR; además, por medios sociales como Twitter, Facebook, blogs, es decir la WEB 2.0. En general la tendencia SoLoMo (Social, Localización, Movilidad) genera gran cantidad de información en forma de mapas, coordenadas, estados, etiquetas, rutas; almacenados en soluciones cloud cada vez con mayor frecuencia y en más organizaciones alrededor del mundo. Para realizar análisis específicos, el *análisis de datos* se clasifica en seis categorías:

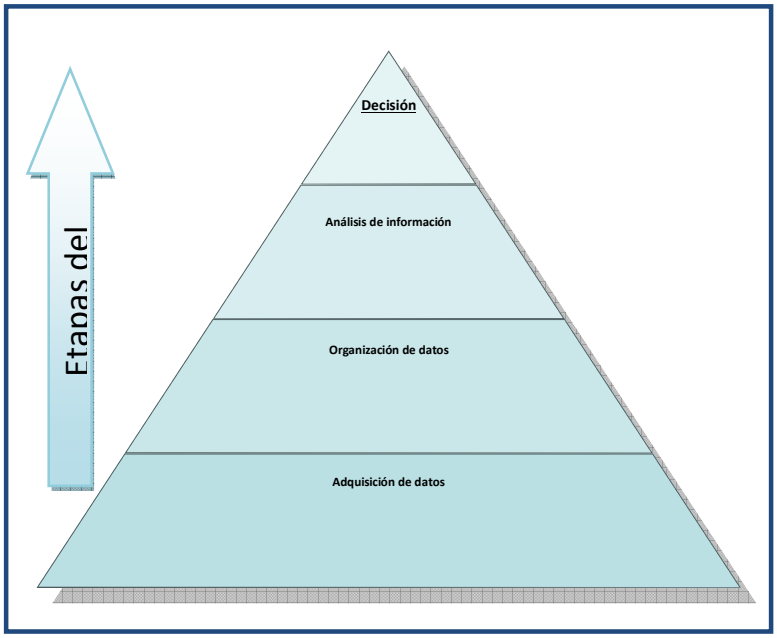


Figura 4. Etapas en el tratamiento de los grandes volúmenes de datos.

2.5.1. Analítica de datos (analytics): es la analítica empresarial o de negocios, ya que analiza datos tradicionales de transacciones y operaciones en organizaciones y empresas.

2.5.2. Analítica WEB: realiza análisis cuantitativo y cualitativo a un sitio WEB, en lo relacionado al tráfico del sitio y los clics realizados por los visitantes en él. Como lo menciona el autor en su obra Analítica WEB: medir para triunfar (Maldonado, 2012) [17], en la página 25, ella analiza y presenta datos recogidos de la WEB para optimizar la estrategia digital de la organización.

2.5.3. Analítica social: realiza análisis de datos a los medios sociales como redes sociales, wikis, blogs, entre otros.

2.5.4. Analítica móvil: realiza análisis a los datos que envían, reciben y transitan por los dispositivos móviles.

2.5.5. Analítica de Big Data: también llamada analítica avanzada, analítica de descubrimiento o analítica exploratoria. Hace referencia al uso de técnicas analíticas aplicadas a conjuntos de grandes volúmenes de datos.

2.5.6. Analítica de M2M (machine to machine): hace referencia a la analítica de datos de máquina a máquina. Internet de las cosas permitirá la interconexión de dispositivos, los cuales capturan datos por medio de sensores, chips de radiofrecuencia entre otros.

La figura 5 presenta una propuesta a la relación entre las categorías de *analítica* del análisis de datos en Big Data.

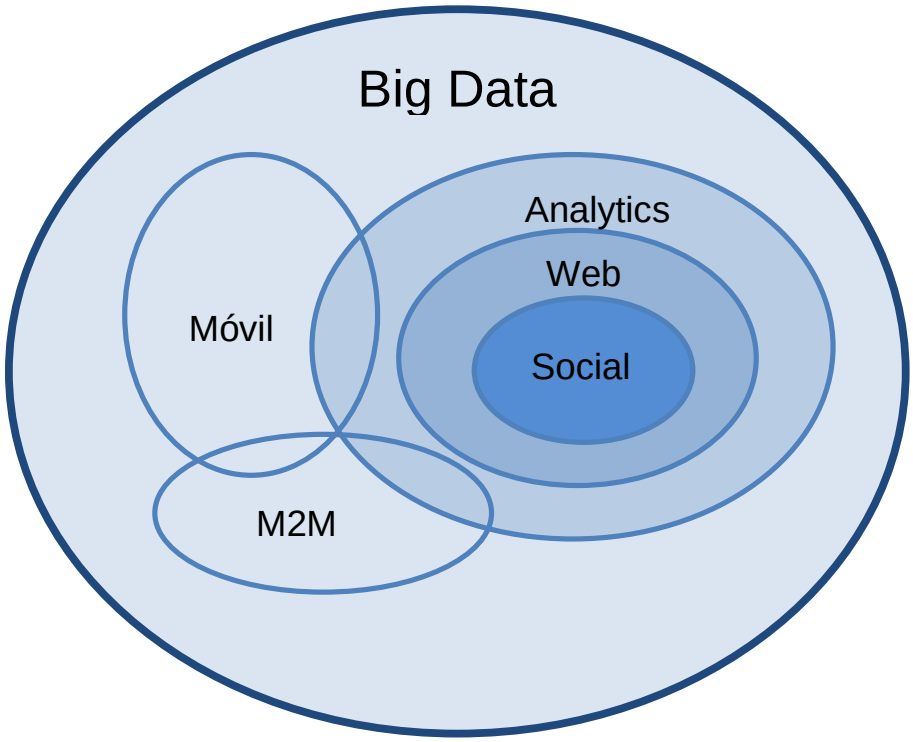


Figura 5. Propuesta a la relación entre las categorías de *analítica* del análisis de datos en Big Data. Fuente: El Autor.

2.6. Analítica social

También llamada análisis social o analítica de medios sociales (social analysis o social media analytics). Es el área de la analítica de datos que realiza análisis de datos a medios sociales; analiza los datos no estructurados que se encuentran en mensajes instantáneos, de correo electrónico, sitios WEB, blogs y en general otros medios sociales a través de diferentes canales, medios de comunicación y dispositivos; para medir y explicar la opinión de los individuos o colectivos frente al tema de observación.

2.6.1. Social Media

O medios sociales, son los datos que se están generando y acumulando procedentes de múltiples fuentes como Twitter, Facebook, Google+, Amazon, chats, blogs, foros entre otros. Corresponden en gran proporción al acceso a Internet por medio de dispositivos móviles. Social media aporta en una proporción grande y en aumento, a la cantidad de datos masivos que se generan a diario. En 2010 Facebook tenía más de 500 millones de usuarios, para John Lovett la participación del cliente en los Social Media se ha vuelto omnipresente (Lovett, 2012) [18].

2.6.2. Herramientas de analítica social

Las herramientas de software para la analítica social se agrupan dentro de las siguientes categorías:

- Herramientas de estadísticas
- Herramientas de analítica
- Herramientas de investigación y monitorización
- Herramientas de análisis de influencia (relevancia)
- Herramientas de reputación

- Herramientas de análisis de actividad en:
    - Redes sociales: Twitter, Facebook, Youtube, entre otras
    - Blogs

Las distintas redes sociales disponen de sus propias herramientas de estadísticas que permiten acceso y análisis de los datos, como por ejemplo:

- **Facebook**: Facebook insights
- **Twitter**: Twitter Web Analytics
- **Google**: Google Analytics (permite crear métricas en redes distintas a las de google), Google Social Hub
- **Youtube**: Youtube analytics

Sin embargo, existen aplicaciones desarrolladas por terceros con el propósito de profesionalizar los datos, con una visión de negocio. Por ejemplo:

- **Twitter**: SocialToo, SocialBro, TweetReach, TweetStats, Twittercounter, Trendistic, Twitalyzer, Crowdbooster, TweetGrader, Retweetrank, Relweetist, Tweetdeck, Seesmic, Hootsuite, Visibly, TwentyFeet y TwitSprout entre muchas otras.
- **Facebook**: Visibly, TwentyFeet

2.6.3. Análisis de sentimientos

El análisis de sentimiento o de sentimientos (sentimental analysis) se conoce también como minería de opinión. Hace referencia al análisis automático del sentimiento para ser representado en indicadores que midan las emociones humanas inmersas en los datos sociales; los cuales pueden provenir de fuentes internas o propias como las almacenadas en CRM corporativos, encuestas a clientes y empleados; o, como las provenientes de fuentes externas y automáticas como rede sociales, blogs, foros, medios de comunicación, entre otros.

El análisis de sentimientos realiza análisis rápido y eficiente, para conocer que se dice de una marca o producto, y, detectar tendencias en internet al respecto; a partir del monitoreo de datos sociales (principalmente redes sociales) provenientes de fuentes internas y externas.

El análisis de sentimientos tiene diferentes tipos de indicadores, por ejemplo positivo, negativo o neutro, bueno, malo o neutro.

Con el posicionamiento del concepto de análisis de sentimientos en software empresarial tipo CRM y especialmente en social media como las redes sociales, su uso y aplicación va creciendo. Por ejemplo, en análisis de sentimientos en contenidos no estructurados se puede medir con tres características:

- Polaridad (en la que el contenido de la opinión de una persona se considera como positiva, negativa o neutra).
- Intensidad (se refiere al grado de emoción que expresa una opinión).
- Subjetividad (la fuente que emite una expresión o comentario es objetiva, parcial o imparcial).

El análisis de sentimientos busca extraer información subjetiva de un texto como el de un mensaje de Twitter o post de un blog, se realiza por medio del procesamiento de lenguaje natural (PLN), la inteligencia artificial o la minería de textos, entre otros.

Para el análisis de sentimientos se pueden utilizar herramientas como Klout y Peer Index, además:

- ❖ Twitalyzer
- ❖ How Sociable
- ❖ Viralheat
- ❖ mBlastmPACT

- ❖ Kred.ly
- ❖ TraacKr
- ❖ Twitrratr
- ❖ Moniter
- ❖ TrendiStic
- ❖ Socialmention
- ❖ Twitter Sentiment

Otras herramientas enfocadas en minería de opinión sobre temas o tópicos específicos son:

- Ciao
- Swotti

2.6.4. Herramientas de investigación y monitorización en social media

Dentro del ámbito de los medios sociales, estas herramientas permiten escuchar, monitorear y obtener datos sobre las actitudes de los usuarios.

Un proyecto de social media debe incluir las etapas de investigación, monitorización y análisis de los datos, ello requiere de tecnologías especializadas para las plataformas de medios sociales. La etapa del análisis es muy importante porque es en ella en donde se puede crear conocimiento y tomar decisiones estratégicas. Sin embargo todas las etapas son complejas e importantes, los datos proceden con variedad diversa; en la monitorización del contenido de los medios sociales se deben obtener informes, cruzar datos, analizar estadísticas entre otros.

Existen herramientas de monitorización para social media, generales y otras específicas para las distintas redes sociales, adecuadas para obtener métricas de monitorización como análisis de influencia, análisis de alcance, métricas internas y externas, métricas de reputación, entre otras.

Algunas herramientas de social media generales son:

- Socilamention
- Google insights
- HowSociable
- Google Alerts
- Kgbpeople
- Social Report
- Radian6
- Twitter Search
- BlogPulse
- Google Analytics
- Hootsuite
- Sproutsocial
- ViralHeat
- UberVu
- Sysomos
- TweetDeck
- Alterian
- BrandMetrics
- BlogScope
- SocialPointer
- Socialseek
- Seesmic

Algunas herramientas de analítica de Web social igualmente utilizadas en analítica de medios sociales son:

- Google Analytics
- Google Social Analytics
- Piwick
- OmnitureSiteCatalyst
- Statcounter
- WoopraAnalytics
- JAWStats
- MochiBot

En cuanto al análisis de actividad en redes sociales (monitorización y análisis) Facebook y Twitter, existen las siguientes herramientas específicas para ello:

En Facebook

- ✓ Conversocial
- ✓ Facebook Grader
- ✓ Facebook Lexicon
- ✓ It`sTrending
- ✓ Social Page Evaluator
- ✓ Open Facebook Search

En Twitter

- ➢ Twitter analyzer
- ➢ Twitalyzer
- ➢ Trendsmap

- Twitter Search
- Locafollow
- TweetReach
- Twitter Counter
- TubeMogul
- Twitter Grader
- TweetStats

Para el análisis de actividad en redes sociales con varios perfiles o cuentas en distintas redes sociales, existen otras herramientas de gestión como las siguientes:

- Hootsuite
- Seesmic
- TweetDeck
- CoTweet

La Tabla 1 contiene en resumen el listado de herramientas y usos, para investigación y monitorización en social media, citadas en el documento.

HERRAMIENTAS	ANÁLISIS DE SENTIMIENTOS	MINERIA DE OPINION	SOCIAL MEDIA	WEB SOCIAL	ACTIVIDAD EN REDES SOCIALES	ACTIVIDAD EN REDES SOCIALES CON VARIOS PERFILES	REPUTACIÓN E INFLUENCIA SOCIAL
Alterian			X		X		
BlogPulse			X		X		
BlogScope			X		X		
BrandMetrics			X		X		
Ciao		X					
Conversocial					X		
CoTweet					X	X	
Crowdbooster					X		

EptisaTI	X						
Facebook Grader					X		
Facebook Lexicon					X		
Google Alerts			X		X		
Google Analytics				X	X		
Google insights			X		X		
Google Social Analytics				X			
Hootsuite			X		X	X	
HowSociable	X		X		X		
Inbenta	X						
It`sTrending					X		
JAWStats				X			
Kgbpeople			X		X		
Klout	X						X
Kred.ly	X						X
Locafollow					X		
mBlastmPACT	X						
MochiBot				X			
Moniter	X						
ObjectiveMarketer					X		
OmnitureSiteCatalyst				X			
Ondore	X						
Open Facebook Search					X		
PeerIndex	X						X
Piwick				X			
Radian6			X		X		
Relweetist					X		
Retweetrank					X		
Seesmic			X		X	X	
Social Page Evaluator					X		
Social Report			X		X		
SocialBro					X		
Socialmention	X				X		
SocialPointer			X		X		
Socialseek			X		X		
SocialToo					X		
Socilamention			X				
Sproutsocial			X		X		
Statcounter				X			
Swotti		X					
Sysomos			X		X		
TraacKr	X						
TrendiStic	X				X		
Trendsmap					X		
TubeMogul					X		
TweetDeck			X		X	X	
TweetGrader					X		

TweetReach					X		
TweetStats					X		
TwentyFeet					X		
Twitalyzer	X				X		X
Twitrratr	X						
TwitSprout					X		
Twitter analyzer					X		
Twitter Counter					X		
Twitter Grader					X		
Twitter Search			X		X		
Twitter Sentiment	X						
Twittercounter					X		
UberVu			X		X		
Viralheat	X		X		X		
Visibly					X		
WoopraAnalytics				X			
Workstreamer					X		

Tabla1. Resumen de herramientas para investigación y monitorización en social media. Propuesta por el autor.

2.7. Arquitectura de Big Data

La arquitectura de Big Data se apoya en componentes que se organizan en torno a capas. La figura 6 presenta la arquitectura de referencia de Big Data propuesta por *Sunil Soares* en 2012.

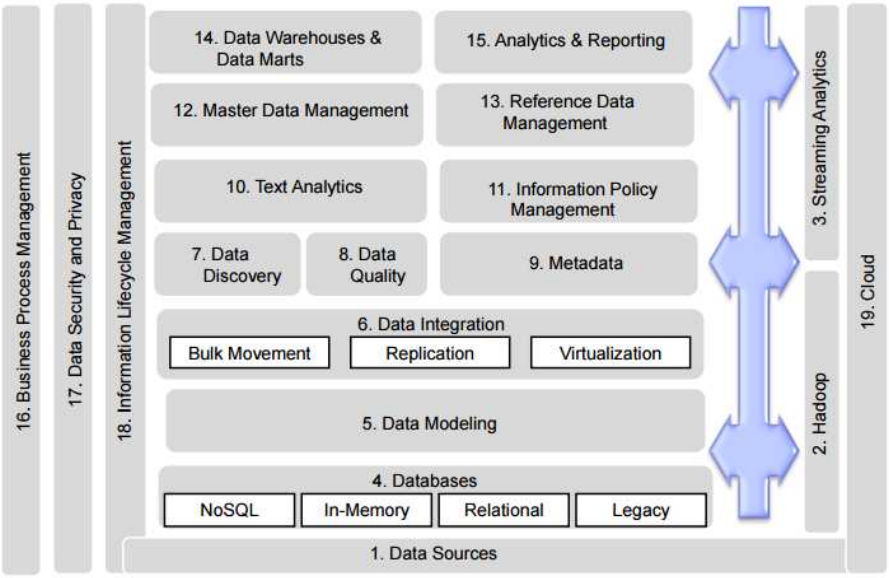


Figura 6. Arquitectura de referencia de Big Data propuesta por Sunil Soares.

La figura 7 presenta la arquitectura de referencia de Big Data, adaptada por Luis Joyanes Aguilar en 2013 con base en la propuesta de Soares en 2012.

2.8. Herramientas teóricas que dan soporte a las tecnologías Big Data

Las tecnologías Big Data se conceptualizan a partir de tres desarrollos tecnológicos, los cuales se formalizaron entre los años 2003 y 2006. Son las tecnologías Google File System (GFS) en 2003, Map Reduce en 2004 y Big Table en 2006, ellas se caracterizan por definir los modelos para la industria de bases de datos en el manejo de grandes volúmenes de datos.

A continuación, en detalle los artículos Google File System (Ghemawat, Gobioff y Leung, 2003) [19], Map Reduce (Dean, Ghemawat, 2004) [20] y Big Table (Chang et al., 2006) [21].

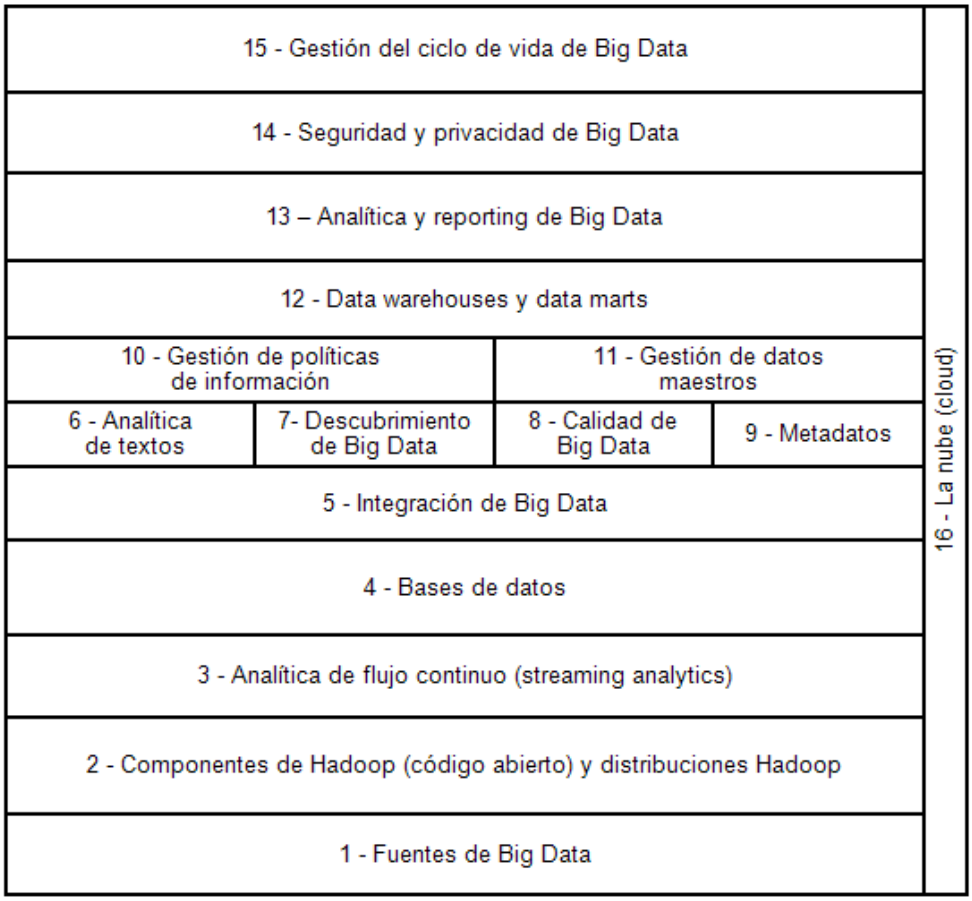


Figura 7. Arquitectura de referencia de Big Data. Fuente Soares (2012:239). Adaptada.

2.8.1. Google File System (GFS)

Introducción

Google File System es un sistema de archivos distribuido escalable para grandes aplicaciones distribuidas intensivas en datos. Proporciona tolerancia a fallos, se ejecuta en hardware de bajo costo, lo que brinda alto rendimiento a un gran número de clientes.

GFS ha sido diseñado e implementado para satisfacer la creciente demanda en procesamiento de datos de Google. El GFS comparte características con anteriores sistemas operativos como escalabilidad, confiabilidad y disponibilidad, sin embargo se han revisado aspectos como los siguientes:

- Debido a que el sistema de archivos se implementa en cientos o miles de computadoras de bajo costo a las que acceden muchos usuarios, son comunes las fallas; además se han observado problemas causados por errores de aplicación, errores de operación del sistema, errores humanos, fallos en el hardware como el disco duro, memoria, distintos conectores, redes y fuentes de poder. Por lo anterior se ha integrado al sistema, monitoreo constante, detección de errores, tolerancia a fallos y recuperación automática.
- Ya que los archivos son enormes para los estándares tradicionales (son comunes archivos de varios GB) y que sea común que los archivos contengan muchos objetos de aplicación tales como documentos web; se hace pesado para manejar *data sets* del orden de terabytes (TB) que contienen billones de objetos comprimidos (del orden de kilobytes), incluso si el sistema de archivos lo soporta. Por lo tanto se han diseñado y revisado hipótesis y parámetros tales como operación y tamaños de bloque en entrada y salida (I/O) de datos.

- La mayoría de los archivos cambian al añadirles nuevos datos en lugar de sobre escribir los datos existentes. Escrituras aleatorias dentro de un archivo son prácticamente inexistentes. Cuando se escribe dentro del archivo, éste queda de lectura únicamente y con frecuencia secuencialmente; variedad de archivos comparten estas características. Algunos pueden ser grandes repositorios que los programas de análisis de datos exploran a través de él. Algunos pueden ser flujos de datos (data streams) continuos generados por aplicaciones en ejecución. Pueden ser datos de archivos. Dado este patrón de acceso en archivos enormes, lo importante para optimizar y mejorar el rendimiento es, anexar y atomizar archivos.
- El co-diseño de la aplicación y de la API del sistema de archivos trae beneficios al sistema dándole flexibilidad. Múltiples *clusters* de GFS son desarrollados para diferentes propósitos. El más grande de ellos tiene más de 1000 nodos de almacenamiento, más de 300 TB de almacenamiento en disco duro y es accesado fuertemente por cientos de clientes en distintas maquinas permanentemente.

Resumen de diseño

Premisas

- Ya que el sistema se construye a partir de componentes de bajo costo, constantemente se debe monitorear, tolerar y recobrar de fallos de componentes de forma rutinaria.
- El sistema almacena un número modesto de archivos grandes, millones de archivos normalmente de 100 MB o de más tamaño, archivos de varios GB también pueden ser manejados con eficiencia por el sistema. Los archivos pequeños son soportados pero no se requiere optimización para ellos.
- La carga de trabajo ante todo consiste de dos tipos de lecturas:

  - Lecturas de grandes s*treaming*: normalmente en operaciones individuales se leen cientos de *kilobytes* o más normalmente 1 MB.
  - Lecturas aleatorias pequeñas: normalmente se leen unos cuantos *kilobytes* en algún desplazamiento arbitrario.

- La carga de trabajo, tiene muchas escrituras secuenciales que agregan datos a los archivos, el tamaño de estas operaciones es similar al de las lecturas de datos; una vez escritos rara vez son modificados nuevamente. Las pequeñas escrituras a posiciones arbitrarias dentro del archivo son soportadas pero no tienen que ser eficientes.
- El sistema debe de manera eficiente implementar semánticas para múltiples clientes que concurrentemente agregan al mismo archivo. Los archivos son usados como productor – consumidor, cientos de productores son ejecutados en distintas maquinas, y concurrentemente agregando a un archivo. Se requiere atomicidad con mínima sobrecarga por sincronización.
- Es más importante el ancho de banda elevado y continuo que la baja latencia, ya que las aplicaciones procesan datos en masa a un ritmo elevado, en contraste pocos requieren tiempos de respuesta rigurosos en operaciones individuales de lectura o escritura.

Interface

GFS proporciona una interface de *filesystem* familiar (no es un API estándar) sin que sea una POSIX (Portable Operating System Interface). Los archivos son organizados jerárquicamente en directorios e identificados por el nombre de su ruta. GFS soporta las operaciones usuales con archivos como crear, eliminar, abrir, cerrar, leer y escribir. Tiene también las operaciones:

  - Instantáneas (*snapshot*): crea una copia de un archivo o carpeta a bajo costo.

- Adicionado: permite a múltiples clientes agregar datos al mismo archivo concurrentemente, garantizando atomicidad en cada operación realizada sin bloqueo adicional al archivo. Éste tipo de archivos son valiosos en la construcción de grandes sistemas distribuidos.

Arquitectura

Un *cluster* GFS se compone de un *single master* (único maestro) y de múltiples *chunkservers* (servidor de pedazos). El cluster es accesado por múltiples clientes, se ejecuta en un servidor Linux en hardware de bajo costo. La figura 8 muestra la arquitectura de Google File System.

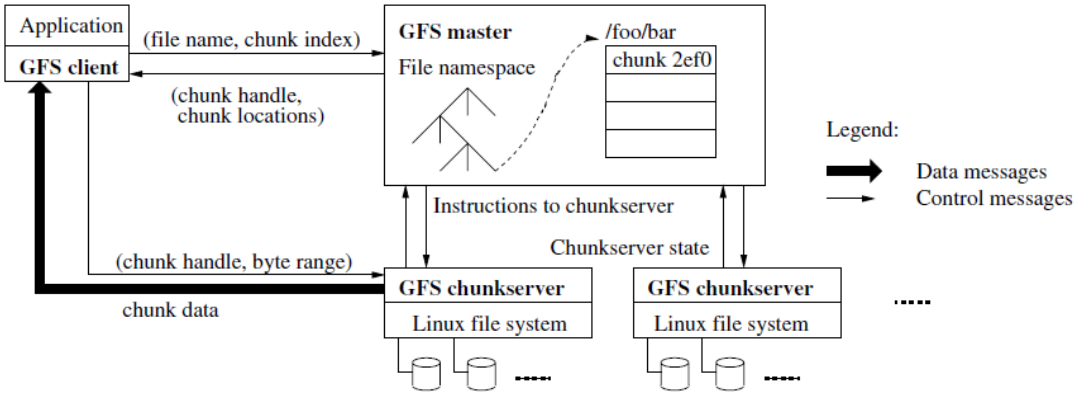


Figura 8. Arquitectura de Google File System (GFS) [19].

Los archivos son divididos en chunks (trozos) de tamaños fijos, cada chunk es identificado por un manejador de 64 bits inmutable y global, el cual es asignado por el maestro al momento de la creación del chunk. El chunkserver almacena chunks en el disco local como archivos Linux. Además lee y escribe datos en el chunk especificado por el manejador del chunk y el intervalo de bytes. Por confiabilidad, cada chunk es replicado en otros chunkservers. Por defecto son almacenadas tres replicas, aunque el usuario puede cambiar éste valor.

El master (maestro) mantiene los metadatos del sistema de archivos como lo es el espacio de nombres (namespace), información de control de acceso, mapeo de los chunks y su ubicación. También controla actividades de todo el sistema como la administración del chunk, recolección de basura de chunks huérfanos, y la migración de chunk entre chunkservers. El master periódicamente se comunica por medio de mensajes con cada chunkserver para dar instrucciones y saber de su estado.

El código del cliente GFS enlazado en cada aplicación implementa la API del sistema de archivos y se comunica con el master y chunkservers para leer o escribir datos en nombre de la aplicación. Los clientes interactúan con el master por metadatos, pero la comunicación de la portadora de datos va directamente al chunkserver.

Ni el cliente ni el chunkserver hace cache a los archivos de datos, ya que la mayoría de las aplicaciones utilizan *streaming* o espacios de trabajo, de gran tamaño. No tenerlos simplifica el cliente y el sistema global porque se eliminan problemas de coherencia del caché.

Los chunkservers no tienen caché en archivos de datos porque los chunks se almacenan como archivos locales y así el búfer cache de Linux mantiene los datos más frecuentes en la memoria.

Master

Tener un solo maestro simplifica el diseño y le permite tener conocimiento del sistema y hacer sofisticadas operaciones de colocación y replicación a los chunks.

Los clientes nunca leen ni escriben datos a través del master. El cliente pregunta al master con qué chunkservers ponerse en comunicación, esta información se guarda en caché para operaciones futuras.

Para realizar la lectura de un archivo, primero el cliente traduce el nombre del archivo y el desplazamiento de bytes especificado por la aplicación en un índice dentro del archivo. Después,

envía al maestro una petición que contiene el nombre del archivo y el índice del chunk. El maestro responde con la información para manejar el chunk y la ubicación de las réplicas. El cliente mantiene esta información en cache utilizando el nombre del archivo y el índice de él. El cliente envía una solicitud a una de las réplicas (a la más cercana). La solicitud especifica el manejador del chunk y una serie de bytes dentro de él. Con respecto al mismo chunk no se requieren más interacciones entre el cliente y el maestro (master) hasta que la información del caché expire o el archivo sea abierto nuevamente. Es común que el cliente solicite varios chunks en una misma petición, el maestro también puede incluir la información de los otros chunks en la siguiente operación de respuesta para evitar futuras interacciones cliente-master sin costo adicional.

Tamaño del chunk

El tamaño del chunk es más grande que el tamaño típico de los bloques en los sistemas de archivos, es de 64 MB, y sus réplicas son almacenadas como un archivo Linux plano en el chunkserver, si es necesario se extiende su tamaño.

Dentro de las ventajas de tener un chunk de gran tamaño, se cuenta el reducido número de interacciones con el master porque la lectura y escritura en el mismo chunk requiere una comunicación inicial con el master para obtener información de ubicación de él. Siendo importante para disminuir la carga de trabajo del sistema si se tiene en cuenta que en su mayoría las aplicaciones leen y escriben de manera secuencial en los archivos. Otra ventaja, es más probable realizar operaciones sobre un chunk de gran tamaño, reduciendo el gasto sobre la red al mantener una conexión TCP con el chunkserver por un periodo de tiempo prolongado. Es también ventaja que se reduce el tamaño de los metadatos almacenados en el master, lo que permite que se mantengan en memoria.

Cuando un archivo es de tamaño pequeño, requiere de pocos chunks, lo que puede ser un inconveniente para el chunkserver si varios usuarios acceden al mismo archivo, pero en la práctica ello no es inconveniente ya que en su mayoría las aplicaciones leen archivos grandes de varios chunks en forma secuencial. Sin embargo, para atender situaciones como ésta, se almacenan ejecutables con un factor de replicación alto para que inicien escalonadamente.

Metadatos

El master almacena tres tipos principales de metadatos: el archivo y nombre del chunk (chunk name spaces), la asignación de archivos a trozos (mappingfrom files to chunks) y la ubicación de la réplica de cada chunk (locations of each chunk's replicas). Los metadatos son mantenidos en la memoria del master. Los dos primeros tipos de metadatos se mantienen de manera persistente ingresando los cambios en un log de operación almacenado en el disco local del maestro y replicado en las maquinas remotas. Este log permite actualizar el estado sencillamente, fiablemente y sin riesgo de inconsistencias en caso de un accidente en el master. La ubicación cada chunk no se almacena de manera persistente en el master, se pregunta a cada chunkserver cuando el master inicia su operación y cada vez que un chunkserver se une al cluster.

Interacciones del sistema

El sistema está diseñado para minimizar la participación del master en todas las operaciones.

Arrendamientos y mutaciones

La mutación es una operación que cambia el contenido o los metadatos de un chunk, como por ejemplo una operación de escritura. Cada mutación se realiza sobre las réplicas del chunk.

Se utilizan contratos (leases) para mantener un orden en las mutaciones para que sea consistente a través de réplicas. El master otorga una chunk lease a una de las réplicas, llamado primario (primary). Éste recoge en orden todas las mutaciones en el chunk. Todas las réplicas siguen este

orden para la aplicación de las mutaciones. Por lo tanto, el orden de las mutaciones se define inicialmente por el orden de arrendamiento elegido por el master, y dentro de un contrato de por los números de serie asignado por el primario. Este mecanismo es diseñado para minimizar la sobre carga en la administración por parte del master

Flujos de datos

El flujo de datos se desacoplo del control de flujo para usar la red de una manera más eficiente. El objetivo es utilizar completamente el ancho banda de cada máquina, evitar los cuellos de botella y la latencia en los enlaces para llevar los datos por la red.

Para utilizar completamente el ancho de banda de cada máquina, los datos se envían linealmente a lo largo de una cadena de chunkservers en lugar de distribuirlos por otra topología, de esta manera el ancho de banda de subida de cada máquina se utiliza completamente para transferir los datos en lugar de ser divididos entre varios destinatarios.

Para evitar los cuellos de botella y enlaces de alta latencia, cada máquina envía los datos al equipo "más cercano" en la red.

Además se minimiza la latencia entubando (pipeline) la transferencia de datos por conexiones TCP. Cada vez que el chunkserver recibe datos, se inicia el reenvío de inmediato. Los pipeline son funcionales porque por medio de la red conmutada se establecen comunicaciones full dúplex. La comunicación de envío y recepción de datos simultáneamente, no afecta las tasas de transferencia.

Al no tener congestión en la red, el tiempo ideal para transferir bytes réplicas es (B/T) + (RL) donde T es el rendimiento de la red y L es la latencia para transferir bytes entre dos equipos o máquinas. Por lo general los enlaces de red son de 100Mbps (T), y L es un valor muy por debajo de 1ms. Por lo tanto, 1 MB puede ser distribuido en aproximadamente 80 ms.

Operaciones atómicas en el grabado

GFS realiza operaciones atómicas en el grabado (atomic record appends) llamadas record append, el cliente únicamente especifica los datos. GFS los anexa al archivo al menos una vez atómicamente (como una secuencia continua de bytes) en un desplazamiento elegido por GFS. Es similar a escribir en un archivo abierto en modo O_APPEND en Unix sin las condiciones de carrera (racecondition) cuando varios escritores lo hacen al mismo tiempo.

Los Record append son utilizados en las aplicaciones distribuidas de google ya que muchos clientes en diferentes máquinas agregan información al mismo archivo y al mismo tiempo.

El cliente envía los datos a todas las réplicas del último chunk del archivo. Se envía el requerimiento al primary, éste verifica que no se exceda el tamaño de 64 MB al grabar el chunk, en caso de ser excedido el tamaño se llena el chunk al tamaño máximo y dice a los secundarios que hagan lo mismo.

Si se presenta un fallo al grabar datos en una réplica, el cliente reintenta la operación.

Las réplicas del mismo chunk pueden contener diferentes datos, no se garantiza que todas las réplicas sean idénticas byte a byte, se garantiza que por lo menos una vez se escriban los datos como una unidad atómica.

Operaciones instantáneas (snapshot)

Los snapshots toman una copia de un archivo o directorio casi instantáneamente, mientras se minimizan las interrupciones de las mutaciones en curso. A menudo se utiliza para crear copias de grandes cantidades de información rápidamente.

Cuando el maestro recibe una solicitud de snapshot, primero revoca cualquier lease pendiente en el chunk de los archivos a punto de intervenir. Esto asegura que cualquier escritura futura a estos chunks necesite interactuar con el master para encontrar la lease. Esto le dará al master la

oportunidad de crear una nueva copia del primer chunk. Los archivos recientemente creados con snapshot apuntan a los mismos chunk como los archivos de origen.

Cuando un cliente quiere escribir en un chunkC después de la operación de snapshot, se envía una solicitud al master para encontrar el lease actual. Los mensajes del master indican que el recuento de referencia para chunkC es mayor que uno. Se pospone la respuesta a la solicitud del cliente y en su lugar recoge un nuevo manejador de chunk para C'. A continuación se solicita a cada chunkserver que tiene una réplica actual de C crear un nuevo chunk llamado C'. Al crear el nuevo chunk en los mismos chunkservers del original chunk C, nos aseguramos de que los datos se pueden copiar a nivel local y no por la red. Desde este punto, el manejo de la solicitud no es diferente de la de cualquier chunk: el master dona una de las réplicas en el nuevo chunkC' y responde al cliente, que puede escribir el chunk normalmente, sin saber que acaba de ser creado a partir de un trozo existente.

Operaciones del master

El master ejecuta todas las operaciones del namespace, además gestiona las réplicas de los chunks en todo el sistema, crea nuevos chunks y sus correspondientes replicas, y coordina diversas actividades dentro del sistema para mantener correctamente las réplicas de los chunk, balancea las cargas en los chunkservers y recupera el espacio de almacenamiento no utilizado.

Gestión de namespace y de bloqueo

Para no agregar lentitud al sistema por cuenta de bloqueos sobre los chunks mientras se realiza operaciones de snapshat, se permiten múltiples operaciones activas y el uso de bloqueos sobre regiones del namespace para garantizar la correcta serialización.

A diferencia de muchos sistemas de archivos tradicionales, GFS no tiene una estructura de datos por directorio que muestra todos los archivos de ese directorio. Tampoco soporta alias para el

mismo archivo o directorio. GFS representa lógicamente su namespace como un mapeo de la tabla de búsqueda a los metadatos. Con la compresión de prefijo, esta tabla se representa de manera eficiente en la memoria. Cada nodo en el árbol del namespace (ya sea un nombre de archivo absoluto o un nombre de directorio absoluto) tiene un bloqueo de lectura-escritura asociado. El bloqueo de lectura en el nombre es suficiente para proteger al directorio principal de eliminación.

Una propiedad de este esquema de bloqueo es que permite que mutaciones simultáneas en el mismo directorio. Por ejemplo, múltiples creaciones de archivos puedan ejecutarse concurrentemente en el mismo directorio: cada uno adquiere un bloqueo de lectura en el nombre del directorio y un bloqueo de escritura en el nombre del archivo. El bloqueo de lectura en el nombre del directorio es suficiente para evitar que el directorio se borre, renombrado o se le realice snapshot. Los bloqueos de escritura en los nombres de archivo serializa los intentos de crear un archivo con el mismo nombre dos veces.

Desde el namespace se pueden tener muchos nodos, objetos de bloqueo de lectura y escritura se asignan “perezosamente” y se borran una vez que no están en uso. Además, los bloqueos se adquieren en un orden coherente para evitar estancamientos: son primero ordenados por nivel en el árbol del namespace y lexicográficamente dentro del mismo nivel.

Ubicación de las réplicas

Un cluster GFS es altamente distribuido. Normalmente tiene cientos de chunkservers repartidos por muchos racks de la máquina. Estos chunkservers se pueden acceder desde cientos de clientes del mismo o diferente rack. La comunicación entre dos máquinas en diferentes racks puede cruzar uno o más conmutadores (switchs) de red. El ancho de banda al interior o fuera del rack puede ser menor que el ancho de banda agregado por todas las máquinas dentro del rack.

La distribución multinivel presenta un desafío único para distribuir datos, por lo relacionado con la escalabilidad, fiabilidad y disponibilidad.

La política de ubicación de la réplica del chunk tiene dos objetivos: maximizar la fiabilidad y disponibilidad de los datos, y maximizar la utilización del ancho de banda de la red. Por tanto, no es suficiente difundir réplicas a través de las máquinas, proteger contra fallos de disco o de la máquina y utilizar totalmente el ancho de banda de cada máquina. También se deben difundir las réplicas de los chunks a través del rack. Esto asegura que algunas réplicas de un chunk sobrevivirán y permanecerán disponibles incluso si todo un rack se daña o queda fuera de línea (por ejemplo, debido a un fallo de un recurso compartido como un switch de red o circuito de eléctrico de potencia).

Creación, re – replicación y rebalanceo

Las réplicas a los chuks se realizan por tres razones:

- Creación de chunks
- Re – replicación a chunks
- Rebalanceo de chunks

Cuando el maestro crea un trozo, elige dónde colocar las réplicas inicialmente vacías.

El master re-replica un chunk tan pronto como el número de réplicas disponible cae por debajo del objetivo especificado por el usuario. Esto puede ocurrir por varias razones: a chunkserver deja de estar disponible, se informa que su réplica puede estar dañada, uno de sus discos está desactivado debido a errores, o la meta de replicación se aumenta. Cada chunk que necesita ser re-replicado se prioriza sobre la base de varios factores. Uno es lo lejos que está de su objetivo de replicación. Por ejemplo, se da mayor prioridad a un chunk que ha perdido dos réplicas que a un chunk que ha perdido sólo una. Primero se re-replican fragmentos de archivos en vivo en

lugar de chunks que pertenecen a los archivos borrados recientemente. Por último, para minimizar el impacto de los fallos en las aplicaciones en ejecución, se aumenta la prioridad a cualquier chunk que está bloqueando el progreso del cliente.

Por último, el master re-balancea las réplicas periódicamente: examina la distribución de la réplica actual y mueve réplicas a un disco de mejor espacio y equilibrar la carga. También a través de este proceso, el master se llena poco a poco de chunk servers, inundándose instantáneamente con nuevos chunks y con el tráfico alto de escritura que viene con ellos. El master también debe elegir qué réplica eliminar. En general, se prefiere eliminar los de chunk servers con espacio libre bajo la media con el fin de igualar el uso de espacio en disco.

Recolección de basura

Después de que un archivo es eliminado, GFS no reclama el espacio físico disponible. Lo hace solamente lentamente durante la recolección de basura, tanto a nivel de archivo y como de chunk. Este enfoque hace que el sistema sea mucho más simple y más fiable.

Cuando se elimina un archivo por la aplicación, el master registra el borrado de inmediato al igual que otros cambios. Sin embargo en vez de reclamar recursos inmediatamente, el archivo se acaba de renombrar a un nombre oculto que incluye la marca de tiempo de borrado. Durante la exploración regular del master de su namespace, se elimina cualquier tipo de archivo oculto si ha existido por más de tres días (el intervalo es configurable).

Hasta entonces, el archivo todavía se puede leer con el nombre especial y no puede recuperarse por el cambio de nombre de nuevo a la normalidad.

Cuando se quita el archivo oculto del namespace, sus metadatos en memoria se borran. Esto rompe sus enlaces con todos sus chunks.

En una búsqueda similar del namespace el master identifica chunks huérfanos (es decir, los que no están accesibles desde cualquier archivo) y borra los metadatos de esos chunks. En un mensaje HeartBeat intercambiado regularmente con el master, cada chunkserver informa de un subconjunto de los chunks que tiene, y el master responde con la identidad de todos los chunks que ya no están presentes en los metadatos del master. El chunkserver es libre de borrar sus réplicas de estos chunks.

Aunque la recolección de basura distribuida es un problema difícil que requiere soluciones complicadas en el contexto de los lenguajes de programación, es bastante simple en GFS. Se identifican fácilmente todas las referencias a chunks: están en los mapeos archivo – chunk a cargo exclusivamente del master.

También se identifican fácilmente todas las réplicas de chunks porque son archivos de Linux bajo directorios designados en cada chunkserver. Cualquier replica que no conoce el master es basura.

El enfoque de la recolección de basura a la recuperación de almacenamiento ofrece varias ventajas sobre eliminación. En primer lugar, es simple y fiable en un sistema distribuido a gran escala en la que los fallos de componentes son comunes. No en todos los casos se tiene éxito al crear un chunk en el chunkserver, dejando réplicas que el master no sabe que existen. Mensajes de eliminación de la reproducción se pueden perder, y el master tiene que acordarse de volver a enviar a través de fallas, tanto por sí mismo y los de los chunkservers.

Detección de réplicas antiguas

Las réplicas de chunks pueden volverse antiguas si un chunkserver falla y pierde mutaciones al chunk mientras se encuentre no disponible. Para cada chunk, el master mantiene un número de versión del chunk para distinguir entre la fecha actual y la antigüedad de la réplica.

El master elimina réplicas antiguas en su recolección de basura regular. Antes de eso, se considera como una réplica antigua que no existe en absoluto cuando no se responde a las peticiones de los clientes para obtener información.

Como otra salvaguardia, el master cuenta con el número de versión del chunk cuando se informa a los clientes que chunkserver lleva a cabo un lease de un chunk o cuando instruye una chunk server para leer el chunk de otro chunkserver en una operación de clonación. El cliente o el chunkserver verifica el número de versión cuando se realiza la operación de modo que está accediendo a los datos siempre hasta a la fecha.

Tolerancia a fallos y diagnostico

La calidad y cantidad de los todos los componentes hacen que ocurran problemas, es más la norma que la excepción: no se puede confiar por completo en las máquinas, ni confiar completamente en los discos duros.

Alta disponibilidad

Entre cientos de servidores de un clúster de GFS, algunos están obligados a estar disponibles en cualquier momento. El sistema general es de alta disponibilidad con dos simples pero efectivas estrategias: recuperación y replicación rápida.

En cuanto a la rápida recuperación, tanto el master y como el chunkserver están diseñados para restaurar su estado y comenzar en segundos sin importar la forma en que terminan. Para GFS no hay una terminación normal y anormal; los servidores se cierran solo con “matar” el proceso. Los clientes y otros servidores sufren entonces un leve contratiempo, ya que el tiempo de espera en sus solicitudes pendientes es corto mientras se conecta al servidor reiniciado.

El número o niveles de replicación pueden ser especificado por el usuario, por defecto el valor es tres. Los masters clonan replicas existentes para mantener cada chunk replicado ya que algún

chunkserver puede estar fuera de línea, o detectar réplicas dañadas a través de la verificación de la suma de comprobación.

El master replica para la fiabilidad de la información, esta información registrada y momentos de chequeo son igualmente replicados en múltiples máquinas. Una mutación en el estado se considera efectuada sólo después de que su registro se ha hecho a disco a nivel local y en todas las réplicas principales. Es decir, los procesos maestros siguen siendo responsable de todas las mutaciones así como actividades en segundo plano, como la recolección de basura que cambia al sistema internamente.

Cuando se produce un error, el sistema puede reiniciar casi al instante. Si su máquina o disco falla, la infraestructura de monitoreo externo GFS inicia un nuevo proceso maestro en otros lugares con el registro de la operación de replicado. Los clientes utilizan sólo el nombre canónico del master, que es un alias de DNS que se puede cambiar si el master se trasladó a otra máquina.

Integridad de los datos

Cada chunkserver utiliza la suma de comprobación para detectar la corrupción de los datos almacenados. Ya que un cluster GFS a menudo tiene miles de discos en cientos de máquinas, experimenta regularmente fallos de disco que causan la corrupción o pérdida de datos tanto en la lectura y escritura. Para el sistema es posible recuperarse de la corrupción mediante otras réplicas en el chunk, pero sería poco práctico para detectar la corrupción mediante la comparación de las réplicas en todos los chunkservers. Por otra parte, las réplicas divergentes pueden ser legales: la semántica de las mutaciones de GFS, en particular, almacenamiento atómico, no garantizan réplicas idénticas. Por lo tanto, cada chunkserver debe verificar de forma independiente la integridad de su propia copia mediante el mantenimiento de las sumas de comprobación.

Un chunk se divide en bloques de 64 KB. Cada uno tiene una suma de comprobación de 32 bits. Al igual que otros metadatos, las sumas de comprobación se mantienen en la memoria y se almacenan persistentemente con en el log, separadamente de los datos del usuario.

Para lecturas, el chunkserver verifica la suma de comprobación (checksum) de los bloques de datos que coinciden con el rango de lectura antes de devolver los datos al solicitante.

Por lo tanto los chunkservers no propagan datos corruptos a otras máquinas. Si un bloque no coincide con la suma de comprobación grabada, el chunkserver retorna un error al solicitante y los informes del fallo con el master. En respuesta, el solicitante leerá de otras réplicas, mientras que el master clona el chunk desde otra réplica. Luego, con una nueva réplica válida, el master instruye al chunkserver que informó la inconsistencia para eliminar su réplica.

La suma de comprobación tiene poco efecto sobre el rendimiento de lectura por varias razones. Dado que la mayoría de las lecturas se realiza al menos a un par de bloques, tenemos que leer y realizar sumas de comprobación sólo a una pequeña cantidad de datos adicionales para su verificación. Las búsquedas de la suma de comprobación y comparación de los chunkserver se realizan sin ningún cálculo de E / S, y la suma de comprobación a menudo se puede superponer con las E / S.

El cálculo de la suma de verificación está muy optimizado para escrituras que anexan al final de un chunk, ya que predominan en la carga de trabajo.

Por el contrario, si una escritura sobrescribe un rango existente del chunk, se deben leer y verificar los primeros y últimos bloques de la gama que se sobrescribirán, a continuación, realizar la escritura, y, finalmente, calcular y registrar las nuevas sumas de comprobación. Si no se verifican los bloques iniciales y finales antes de sobrescribir, las nuevas sumas de comprobación pueden ocultar la corrupción que existe en las regiones que no se han sobrescrito.

Durante los períodos de inactividad, los chunkservers escanean y verifican el contenido de chunks inactivos. Esto nos permite detectar la corrupción en chunks que rara vez se leen. Una vez que se detecta la corrupción, el master puede crear una nueva réplica sin corrupción y eliminar la réplica dañada. Esto evita que un chunk inactivo pero corrupto engañe al master al hacer creer que tiene suficientes réplicas válidas de un chunk.

Herramientas de diagnóstico

El registro de diagnóstico extenso y detallado contribuye al aislamiento de problemas, depuración y análisis de rendimiento, incurriendo en un costo mínimo. El registro de diagnóstico extenso y detallado ha ayudado inmensamente en el aislamiento de problemas, depuración y análisis de rendimiento, mientras que incurrir en solamente un costo mínimo.

Sin registros (logs), es difícil de entender las interacciones entre las máquinas. Los servidores GFS generan registros de diagnóstico que registran muchos eventos importantes (como chunkservers que van caen y se levantan) y todas las peticiones RPC y respuestas. Estos registros de diagnóstico se pueden borrar libremente sin afectar a la exactitud del sistema.

Los registros de RPC incluyen las solicitudes exactas y respuestas enviadas por el cable, a excepción de los datos del archivo que se leen o escriben.

Al asociar peticiones con respuestas y cotejar los registros de RPC en diferentes máquinas, es posible reconstruir toda la historia de las interacciones para diagnosticar un problema. Los registros también sirven como huellas para las pruebas de carga y análisis de rendimiento.

El impacto en el rendimiento por el registro, es mínimo (y ampliamente superado por los beneficios), ya que estos registros se escriben secuencialmente y de forma asíncrona. Los acontecimientos más recientes también se mantienen en la memoria y disponibles para el monitoreo en línea.

2.8.2. MapReduce

En el artículo “MapReduce: procesamiento simplificado de datos sobre clusters de gran tamaño” escrito por Jeffrey Dean and Sanjay Ghemawat en 2004, se define a MapReduce como un modelo de programación y una implementación para el procesamiento y generación de grandes data sets, en el que el usuario especifica una función *map* que procesa un par clave/valor (key/value) para generar un set intermedio de pares clave/valor, y una función reduce que mezcla todos los valores intermedios asociados con la misma clave intermedia.

Los programas realizados son automáticamente paralelizados y ejecutados en los clusters construidos en máquinas de bajo costo (comodity machines). El run-time del sistema se encarga de particionar los datos de entrada, agendar la ejecución de los programas en un conjunto de máquinas, administrar los fallos en las máquinas y manejar la comunicación interna entre las máquinas. Lo anterior brinda una gran oportunidad a los programadores sin experiencia en sistemas paralelos y distribuidos, para aprovechar los recursos de un gran cluster de procesamiento distribuido.

MapReduce es una implementación que se ejecuta en un cluster construido con máquinas de bajo costo, es altamente escalable. Puede procesar varios terabytes de datos en miles de máquinas. Diariamente el cluster de Google ejecuta miles de trabajos enviados por cientos de programas implementados por personas que encuentran en él un sistema fácil de utilizar.

Introducción

MapReduce es una abstracción que permite:

- ✓ Ocultar detalles de paralelización
- ✓ Tolerancia a fallos

- ✓ Distribución de datos
- ✓ Balanceo de cargas

Lo anterior en una librería; como la solución a la dificultad encontrada anteriormente (inicio del 2000) en Google para procesar grandes cantidades de datos en bruto, ya que distribuir los datos por miles de máquinas, paralelizar la computación, distribuir los datos y manejar los fallos tomaba más tiempo y esfuerzo que realizar los cálculos sencillos sobre los grandes volúmenes de datos.

Esta abstracción es inspirada en las primitivas *map and reduce* de muchos lenguajes funcionales entre ellos *Lisp*. Básicamente consiste en realizar una operación *map* a cada registro para obtener un par intermedio key/value, luego, se realiza la operación *reduce* a todos los valores que comparten la misma clave a fin de combinar los datos obtenidos apropiadamente. Éste modelo funcional permite paralelizar grandes cálculos de forma fácil.

Modelo de programación

El cálculo toma como entrada un par de valores key/value, y produce un par de valores de salida key/value. El usuario de la librería MapReduce expresa el cálculo como dos funciones Mapa and Reduce.

Ambas funciones, Map and reduce son escritas por el usuario.

La función Map toma como entrada pares y produce un conjunto intermedio de pares key/values. La librería MapReduce junta todos los valores intermedios asociados con la misma key intermedia y las pasa a la función Reduce.

La función Reduce recibe la key intermedia y un conjunto de valores para esta key, se juntan y combinan estos valores para conformar un conjunto más pequeño de valores. Normalmente, ninguno o un valor es el resultado del llamado a la función Reduce. Los valores intermedios se

suministran a la función Reduce por medio de un iterador, con lo que se permite manejar listas de valores que son muy grandes para estar en memoria.

Implementación

MapReduce permite diferentes implementaciones de su sistema, ello depende del tipo de hardware con que se cuente, por ejemplo:

- Maquinas pequeñas con memoria compartida
- Computadora de múltiples procesadores
- Cluster de equipos de bajo costo conectadas por un switch ethernet

Google implementó un cluster de equipos de bajo costo conectadas por un switch Ethernet con las siguientes características:

- Maquinas con procesadores dual core x86, con sistema operativo Linux y con 2 GB o 4 GB de memoria RAM en cada máquina.
- Switch's Ethernet de bajo costo, de 100 Mbps y de 1Gbps.
- Cientos o miles de máquinas para conformar el cluster, con fallas frecuentes.
- Discos duros IDE de bajo costo, conectados directamente a las máquinas, con un sistema de archivos creado por Google.

La ejecución de MapReduce se lleva a cabo en varias máquinas (procesamiento en paralelo), ya que se particiona automáticamente la entrada principal de datos en M partes. Los llamados a la función Reduce son distribuidos por una función que particiona los conjuntos intermedios de pares key/value. Tanto el número de particiones como de funciones de partición son definidas por el usuario.

La figura 9 presenta la información general de ejecución, enseñando de forma general de operación de MapReduce, en la tabla 2 se brinda una breve explicación al esquema y a la numeración que ella incluye.

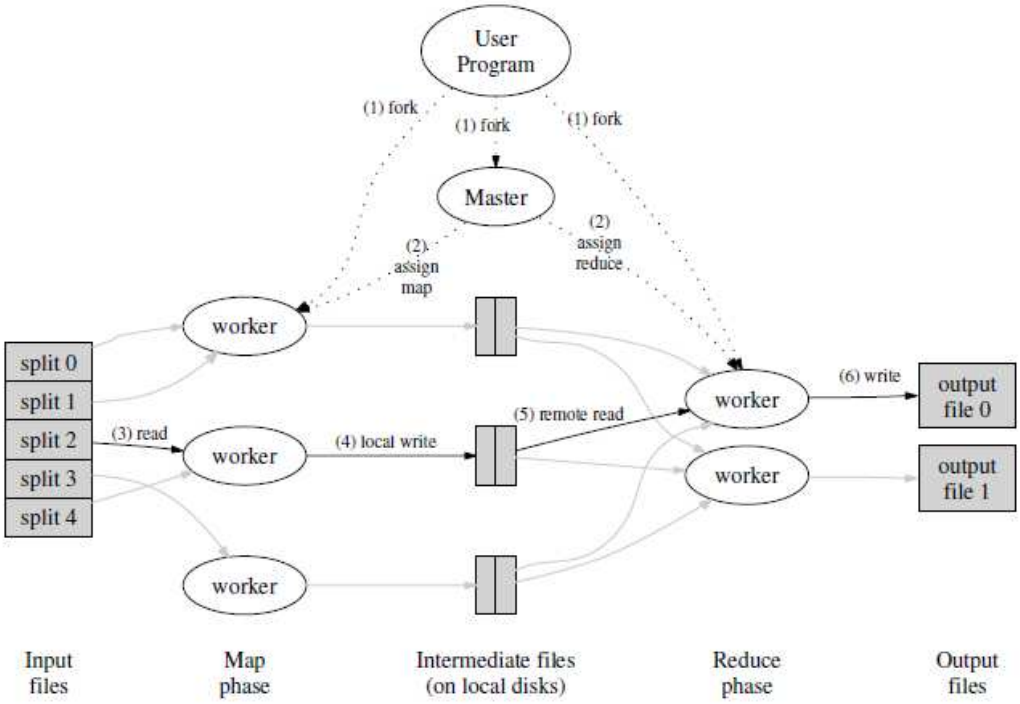


Figura 9. Información general de ejecución de MapReduce.

Acción	Descripción
1	La librería MapReduce del programa del usuario fracciona los archivos de entrada en M partes, normalmente en tamaños entre 16MB y 64MB definidos por el usuario dentro de los parámetros. Luego se inicia la ejecución de muchas copias del programa en los equipos del cluster.
2	El master es la copia más importante del programa, las demás copias trabajan para el Maestro. Hay M tareas y R tareas Reduce para asignar. El Maestro asigna a cada trabajador una de las tareas M o una tarea Reduce.
3	Al trabajador se le asigna una tarea M y lee el contenido de ella. Analiza los

	pares *key/value* y pasa cada par a la función *Map* definida por el usuario. El par intermedio *key/value* producido por la función *Map* es mantenido en memoria.
4	Periódicamente, los pares en memoria son escritos en disco local, fraccionados en R partes por una función. La ubicación en disco local de estos pares es entregada al master, quien a su vez es responsable de reenviar estas ubicaciones a los trabajadores Reduce.
5	Cuando un trabajador Reduce es notificado por el Master sobre una ubicación, él hace llamadas a procedimientos remotos para leer los datos desde el disco local del trabajo Map. Luego que Reduce lee todos los datos de los pares intermedios, los ordena por el campo key, agrupando a todas aquellas key iguales. Si la cantidad de datos intermedios es muy grande para estar en memoria, se hace necesario ordenarlos externamente.
6	Los trabajos Reduce operan sobre todos los pares intermedios ordenados. La respuesta de la función Reduce es adicionada al final del archivo de salida de cada partición Reduce.
7	Cuando todas las tareas Map y Reduce han terminado, el Master retorna al programa del usuario, luego MapReduce llama al programa de usuario retornado en el código del usuario.

Tabla 2. Descripción de las acciones de ejecución de MapReduce

En cuanto al Master, mantiene varias estructuras de datos. Para cada tarea Map y Reduce almacena alguno de los siguientes estados:

- Inactivo
- En progreso
- Terminado

Además la identidad de la maquina si se encuentra con una tarea terminada o en progreso.

El Master envía periódicamente comandos ping a cada trabajador, marcándolos como trabajos fallidos si no escucha respuesta de ellos dentro de un cierto tiempo. Luego que una tarea se

termina, el trabajador toma nuevamente el estado Inactivo, con lo que queda disponible para un trabajo futuro.

Cuando una tarea es ejecutada por primera vez por un trabajador A y luego ejecutada por un trabajador B ante la falla del trabajador A, todos los trabajadores son notificados de esta re-ejecución para que lean los datos del trabajador B y no del trabajador A.

Por medio de *chekpoints* periódicos se reestablecen las tareas del Master en el caso que éste falle, los clientes pueden detectar un fallo del Master y determinar si inician nuevamente con su operación.

Cada tarea Reduce en progreso escribe su salida en un archivo temporal. Una operación Map produce R archivos (uno por cada operación Reduce). Así, una vez la operación Map termina, el trabajador envía un mensaje al Master en el que se incluyen los nombres de los R archivos temporales, los cuales son grabados en la estructura de datos del Master. Cuando la operación Reduce termina, el trabajador automáticamente renombra el archivo temporal de salida.

Ya que dentro del cluster los datos de entrada son manejados por el sistema de archivos de Google (Google File System- GFS) porque se almacenan en disco local, y así el ancho de banda del sistema no se utiliza intensivamente. GFS divide cada archivo en bloques de 64 MB y almacena tres copias (por defecto) de cada bloque en diferentes máquinas. El Master de MapReduce tiene la información de la ubicación de los archivos de entrada, con ellos agenda tareas Map a las máquinas que contengan una réplica de estos datos, con el fin de tener en la misma máquina los datos y el procesamiento, logrando maximizar también el uso del ancho de banda dentro del sistema.

Dentro del proceso MapReduce, Map se divide en M piezas, así mismo Reduce en R. Es normal que se tengan más piezas M y R que máquinas (trabajadoras) en el cluster, sin embargo las

maquinas trabajadoras favorecen el balanceo de carga dentro del sistema al estar realizando varias tareas, también favorece la tolerancia a fallos en caso de que un trabajador falle porque otro trabajador tomara su trabajo. A menudo M=20000, R=5000 usando 2000 máquinas trabajadoras. R es un número mucho menor ya que la salida del archivo a reducir debido a la salida de cada tarea a reducir termina en un archivo de salida separado.

En MapReduce hay operaciones que toman más tiempo de lo normal, a estas operaciones se les denomina operaciones rezagadas. Pueden ocurrir por distintos factores, es apenas normal que dentro de un cluster de gran tamaño sus componentes físicas no funcionen bien en todo momento, por ejemplo un disco duro dañado puede producir que la corrección de errores retrase el proceso, lo que a su vez va a retrasar tareas previamente agendadas, incluso aumentando la competencia por el uso de recursos como CPU, memoria RAM, disco duro y ancho de bando en las comunicaciones. Para contrarrestar estas dificultades, MapReduce cuenta con una estrategia que ha permitido disminuir en un 44% el tiempo que tardan las operaciones MapReduce en realizarse. Consiste en que cuando una tarea está a punto de terminar, el Master inicia una copia de seguridad con el estado del proceso y de lo que falta por realizar. Luego la operación se da por terminada cuando todas sus tareas se han realizado.

2.8.3. Bigtable

Con el artículo “Bigtable: un sistema de almacenamiento distribuido para datos estructurados”, como referencia, se realizan las siguientes consideraciones acerca del tema.

Bigtable es un sistema de almacenamiento distribuido para el manejo de datos estructurados a escalas muy grandes, del orden de peta bytes, implementado en miles de servidores de bajo costo. Google ha desarrollado varios de sus productos utilizando Bigtable entre ellos Google Analytics, Google Earth, indexación WEB y Google finance.

Introducción

Bigtable es diseñado para escalar de manera confiable petabytes de datos y miles de máquinas, logrando varias metas:

- Tener amplia aplicabilidad
- Escalabilidad
- Alto rendimiento
- Alta disponibilidad

Google utiliza a Bigtable en aproximadamente sesenta de sus productos, como Orkut, búsquedas personalizadas, Google Analytics, Google Earth, indexación WEB, Google finance, Writely entre otras. Utilizan éste sistema para distintas cargas de trabajo como el procesamiento por lotes orientado al rendimiento sensible a la latencia de los datos de los usuarios finales.

Los cluster Bigtable permiten distintas configuraciones que les permiten almacenar varios cientos de tera bytes de datos; es similar a una base de datos ya que tienen en común estrategias de implementación como bases de datos paralelas, en memoria, escalabilidad y alto rendimiento; sin embargo Bigtable proporciona una interface diferente a las de estos sistemas.

Bigtable no es compatible con un modelo de datos relacional, pero soporta un modelo sencillo que soporta control dinámico sobre el diseño de datos y el formato. Los datos son indexados utilizando nombres de filas y columnas que pueden ser cadenas arbitrarias, trata los datos como cadenas sin interpretar, aunque permite organizar los contenidos de datos estructurados o semi-estructurados en estas cadenas. Los clientes pueden configurar (de forma cuidadosa) la ubicación de sus datos y controlar dinámicamente si los datos se toman de la memoria o del disco duro.

Modelo de datos

Bigtable es un mapa ordenado multidimensionalmente, se caracteriza por ser disperso, distribuido y persistente.

El mapa se indexa por fila, columna y una marca de tiempo. Cada valor del mapa es una matriz de bytes sin interpretar.

Las filas dentro de la tabla, son cadenas arbitrarias. Cada lectura o escritura de datos en una sola fila es atómica, independientemente del número de columnas diferentes leídas o escritas en la fila.

Los datos se encuentran ordenados lexicográficamente por fila. El rango de la fila de la tabla es particionado dinámicamente, a su vez cada rango de fila es llamado tablet, la cual es la unidad de distribución y balanceo de carga. Se obtiene como ventaja con la lectura de rangos de filas pequeños que requiere comunicación con un pequeño número de máquinas. Los clientes pueden aprovechar esta ventaja seleccionando una buena ubicación para acceder a sus datos.

Las columnas son agrupadas en conjuntos llamadas familias de columnas, las cuales forman la unidad básica de control de acceso. Los datos almacenados en una familia de columnas normalmente son del mismo tipo. La familia de columnas debe ser creada antes de almacenar datos en ella.

Cada celda en una Bigtable puede contener múltiples versiones del mismo dato, estas versiones son indexadas por marca de tiempo (timestamp), las cuales son enteros de 64 bits. Ellos pueden ser asignados por Bigtable en cuyo caso estaría representando el tiempo real. De esta manera las distintas marcas de tiempo son almacenadas y ordenadas en orden decreciente para que la versión más reciente se la que se lea por defecto. El cliente puede especificar el número de

versiones previas para una celda, el recolector de basura se encargará de tener únicamente los valores configurados.

API de Bigtable

La API de Bigtable ofrece funciones para crear y eliminar tablas y familias de columnas. También ofrece funciones para cambiar en el cluster, tablas, familias de columnas y metadatos, además el control de acceso a los datos.

Las aplicaciones cliente pueden escribir o borrar valores en la Bigtable, buscar valores de filas individuales, o utilizar subconjuntos de datos de una tabla.

Bigtable soporta otras características que permite al usuario manipular datos de forma más compleja. Entre ellas:

- Transacciones de una sola fila
- Celdas como contadores enteros
- Scripts de clientes en los espacios de dirección de los servidores (los scripts se deben escribir en Sawzall [22])

Bigtable puede ser utilizado con MapReduce por medio de un framework desarrollado por Google que ejecuta cálculos en paralelo a gran escala

Bloques de construcción

Bigtable está construido sobre la infraestructura de Google, por ejemplo utiliza la infraestructura de Google File System (GFS) para los archivos de datos y de registro (logs).

Bigtable como sistema distribuido, se aloja sobre un sistema operativo el cual se encarga de manejar el cluster y otros procesos distribuidos de otras aplicaciones, así como la programación de tareas, gestión de recursos compartidos, fallos en el cluster y seguimiento a todo el sistema.

Google SSTable (Sorted String Table) es un formato usado internamente para almacenar datos en Bigtable, una de sus características más importantes es que permite mapear la tabla en memoria lo que permite realizar búsquedas sin tocar el disco duro, sin embargo si es necesario, luego se obtiene del disco duro el bloque apropiado de datos.

SSTable ofrece persistencia, mapa de ordenamiento inalterable en los pares de tipo cadenas key/value las cuales son valores arbitrarios, internamente cada bloque SSTable tiene 64 KB de tamaño por defecto pero este valor es configurable. Se utiliza un bloque de índices para ubicar bloques, éste bloque se encuentra al final de la SSTable; el índice se carga en memoria al abrir la SSTable.

Bigtable cuenta con un servicio distribuido, persistente, de alta disponibilidad llamado Chubby, éste servicio mantiene cinco réplicas de datos, una de ellas es elegida como el master para atender distintos tipos de requerimientos relacionados, este servicio se mantiene cuando la mayoría de las réplicas se están ejecutando y se comunican entre sí. Las lecturas y escrituras a los archivos son atómicas, utilizan un seguro, la librería Chubby del usuario permite almacenamiento cache.

Cada usuario mantiene una sesión Chubby, la cual expira luego de pasado cierto tiempo sin que sea renovada. Si la sesión del cliente expira se pierde el seguro y se inicia el procedimiento para notificar la expiración de la sesión y detener las correspondientes llamadas de la sesión.

Bigtable utiliza Chubby para las siguientes tareas:

- Asegurar que al menos un master este activo todo el tiempo.
- Almacenar la ubicación de inicio de los datos dentro de la Bigtable.
- Para descubrir servidores de tablets y terminar con aquellos servidores fuera de servicio.
- Para almacenar la información de la familia de columnas de cada tabla.

- Almacenar la información de la lista de control de acceso.

Chubby es vital para Bigtable, ya que si se detiene su funcionamiento por un espacio prolongado de tiempo, también se detiene Bigtable.

Implementación

Los componentes principales de la implementación de Bigtable, se muestran en la figura 10.

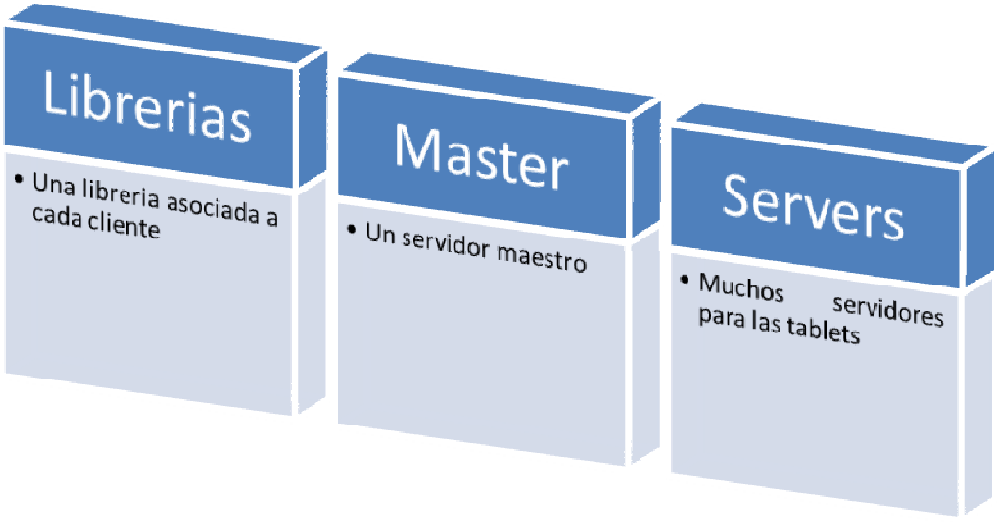


Figura 10. Componentes principales de la implementación de Bigtable. El autor.

El master es el encargado de la asignación de tablets al servidor de tablet, detectar la adición y expiración de servidores de tablets, balancear la carga del servidor de tablets, recolectar la basura de los archivos en GFS y se encarga de los cambios en el esquema como la creación de tablas y familias de columnas.

Cada servidor de tablets maneja un conjunto de tablets entre diez y mil tablets; además, maneja los requerimientos de lectura y escritura de las tablets que se han cargado y divide aquellas tablets que han crecido demasiado.

Los servidores de tablets pueden ser adicionados o eliminados del cluster (dinámicamente) para configurar adecuadamente las cargas de trabajo.

Bigtable es un sistema distribuido de almacenamiento con un solo master. Es importante destacar que los datos del cliente no pasan por el master. Los clientes se comunican directamente con los servidores de tablets, por lo tanto el master coordina las operaciones pero no opera el gran volumen de datos del usuario.

De manera general, un clúster de Bigtable almacena un número de tablas, cada tabla maneja un conjunto de tablets y cada tablet contiene los datos en un rango de filas. Cuando el sistema se inicia, consta en una sola tabla, pero al crecer se divide automáticamente en múltiples tablets, cada una con un tamaño que varía entre 100 MB a 200 MB en la configuración por defecto.

Composición de una tabla (tablet)

La figura 11 muestra la jerarquía de tres niveles que tiene una tabla para almacenar información de la ubicación de las tablets.

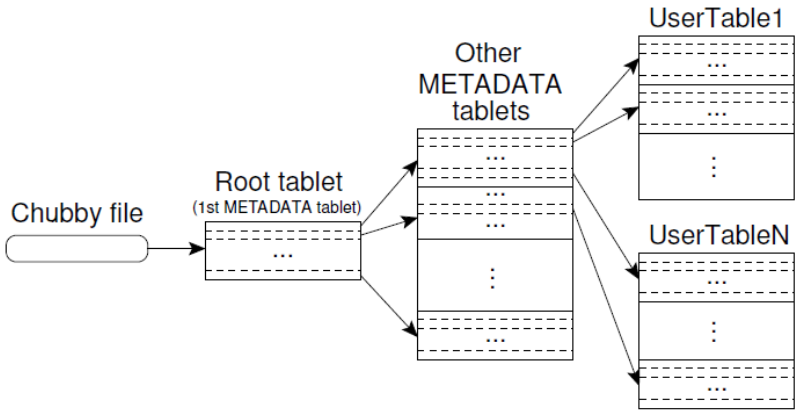


Figura 11. Jerarquía y composición de una tabla en Bigtable.

El primer nivel es un archivo almacenado en Chubby que contiene la ubicación de la tabla raíz o principal (root tablet). La tabla raíz (root tablet) contiene la ubicación de todas las tablas, entre ellas una muy importante: METADATA.

Las tablas METADATA contienen la ubicación de un conjunto de tablets pertenecientes a los usuarios. Pero, la tabla raíz es la primera, es especial porque no se divide como las demás tablas, ello garantiza una jerarquía de tres niveles.

La tabla METADATA almacena la ubicación de una tabla, cada fila almacena aproximadamente 1 KB de datos en memoria. Cada tabla METADATA tiene un límite de 128 MB, el esquema propuesto en Bigtable con tres niveles de jerarquía direcciona hasta 2^{34} tablas.

En la tabla METADATA se almacena otro tipo de información, la de logs que permite registrar eventos relacionados con cada tabla como por ejemplo cuando un servidor inicia su servicio; información como esta es permite analizar el rendimiento del sistema y realizar depuración del mismo.

Asignación de tablas

Cada tabla es asignada por un servidor de tablets a la vez. En esta parte es importante el papel del master ya que tiene la información del conjunto de tablas asignadas asociadas a sus correspondientes servidores, además el registro de aquellas tablas que están sin ser asignadas.

Bigtable utiliza Chubby para realizar el seguimiento a los servidores de tablets, cuando inicia operación un servidor de tablets se crea un seguro con un único nombre en un directorio Chubby. Los master monitorean este directorio en busca de servidores para las tablets, si un servidor de tablets pierde el seguro se detiene el servicio hacia esta Tablet, ello puede ocurrir por cambios en la red de datos que causen la cancelación de la sesión con el servicio Chubby. En cuanto a Chubby, este brinda un mecanismo que permite al servidor de tablets determinar si aún cuenta con el seguro sobre cada Tablet. Ello se realiza sin incurrir en tráfico para la red. En caso de haber perdido el seguro, el servidor intentara tomar nuevamente sus tablas, pero si ellas ya no existen el servidor se auto-destruye.

El master por su lado, es responsable de detectar cuando un servidor de tablets no está sirviendo a sus tablets para reasignarlas tan pronto como sea posible.

El master solicita periódicamente a los servidores de tablets por el estado de sus seguros para determinar si es necesario reasignarlas a otros servidores de tablets.

Para garantizar que Bigtable no sea vulnerable a los problemas derivados de la red entre el master y Chubby, el master se elimina a si mismo si su sesión con Chubby expira. Sin embargo, los errores del master no cambian la asignación de tablets a los servidores de ellas.

Cuando el manejador del sistema del cluster inicia al master de Bigtable, se realiza el siguiente proceso:

1. El master toma un seguro único en Chubby que impide instanciaciones concurrentes del master.
2. El master busca el directorio de servidores en Chubby para determinar con que servidores cuenta.
3. El master se comunica con cada servidor de tablets para conocer que tablets están asignadas a que servidores.
4. El master estudia la tabla de METADATOS para conocer como están las tablets.

Si durante la realización del procedimiento anterior se encuentra una tablet sin ser asignada, el master la agrega al listado de tablets sin ser asignadas, y próximas a asignación.

El conjunto de tablets existentes solo cambia cuando una tablet es creada o eliminada, o cuando una tablet se divide en dos debido a su gran tamaño, o cuando dos tablets pequeñas se fusionan para conformar una sola. Estos cambios son seguidos por el master. El caso de dividir una tabla en dos es tratado de manera espacial ya que es un proceso que inicia el servidor de tablets, este

cambio es informado a la tabla de METADATA para la correspondiente grabación de la información para luego notificar al master.

Persistencia de las tablas

GFS es el encargado de la persistencia. La figura 12 muestra la representación de una tabla.

Las actualizaciones se confirman en un registro que almacena como se ha realizado la grabación de los datos. Las operaciones recientemente realizadas son almacenadas en memoria en un buffer llamado memtable y las más antiguas se almacenan en SSTables.

Para grabar una tabla, un servidor de tablets lee los metadatos de la tabla METADATA, la cual contiene el listado de SSTables que componen la tablet y los puntos de restauración (los que son punteros a los log que tienen información de la tabla). El servidor lee los índices del SSTable en la memoria y reconstruye la memtable para realizar todas las actualizaciones que se indiquen en los puntos de restauración.

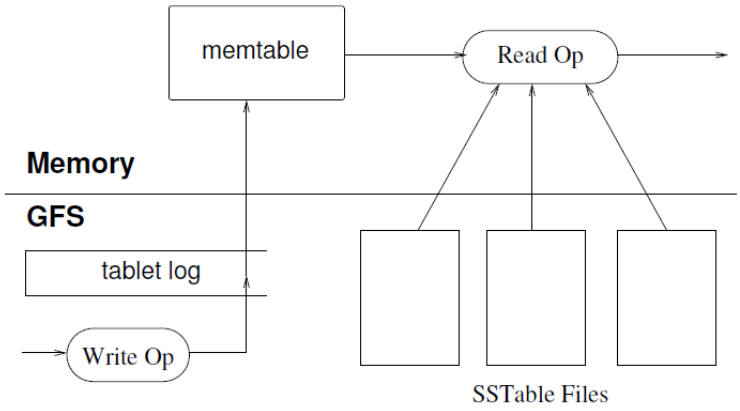


Figura 12. Representación de una tabla en GFS.

Una operación de escritura, se inicia con una petición al servidor de la tablet, el cual verifica que el remitente este autorizado dentro del listado manejado por Chubby, y así proceder a realizar el cambio solicitado y de esta manera el cambio es escrito en el log y el contenido insertado en el memtable.

En el caso de la operación de lectura, la petición al servidor de tablet se verifica al remitente y que tenga autorización. Las lecturas aprobadas son una vista combinada de las Estables y la memtable, ellas organizan lexicográficamente las estructuras de datos para ser presentadas como información al usuario.

Las operaciones de lectura y escritura se pueden realizar a pesar de las operaciones internas de división y unión de las tablas.

Compactación de tablas

Cada operación de escritura aumenta el tamaña de la memtable, y al alcanzar un tamaño determinado (umbral) esta se congela, para que sea creada una nueva. La tabla congelada es convertida en una SSTable para ser escrita en GFS, a este proceso se le conoce como compactación menor.

El proceso de compactación menor tiene dos objetivos:

- Reducir el uso de memoria del servidor de tablets.
- Reducir la cantidad de datos a leer desde el log de operaciones en caso que el servidor haya dejado de funcionar.

Las operaciones de escritura y lectura pueden continuar mientras el proceso de compactación de tablas ocurre simultáneamente.

El proceso de compactación mayor es el proceso en el que se reescriben varias SSTables en una sola. Un ciclo de verificación de datos de Bigtable revisa las nuevas tablas SSTables en busca de liberar recursos de almacenamiento dentro de estas tablas, de esta manera los datos eliminados salen del sistema de almacenamiento.

Capítulo 3
Técnicas y Tecnologías

3.1 Minería de datos

La minería de datos es un proceso no elemental para encontrar relaciones, correlaciones, dependencias, asociaciones, modelos, estructuras, tendencias, clases, segmentos, en grandes conjuntos de datos, los cuales están almacenados en bases de datos relacionales o no relacionales; mediante métodos matemáticos, estadísticos o algorítmicos, trabajando de manera automatizada.

Una definición un poco más reducida sobre minería de datos, apunta hacia las técnicas. Son técnicas para análisis de datos, adaptada a los grandes datos para extraer conclusiones o información relevante de ella. Estas técnicas hacen uso de: clasificación, regresión, reglas de asociación y análisis de clúster.

En lo relacionado al contenido WEB, la minería de datos se clasifica en minería WEB y minería de textos.

El uso extendido de la sociedad de blogs, wikis, etc., plantea el reto de analizar la gran cantidad de información que se genera en la WEB, clasificando en dos tipos la minería para ésta clase de contendidos en minería social y minería de opinión.

No hay una única o definitiva clasificación para la minería de datos, ya que puede haber clasificaciones a partir de las fuentes de la información, como por ejemplo la proveniente de sensores conectados por internet (internet de las cosas) o la WEB, o por las técnicas para el análisis de datos, es decir regresión, clasificación, entre otras. Inclusive se debate, si considerar

la minería de datos como estadística aplicada o como una solución computacional. A pesar de lo anterior, sin duda la estadística y la computación se unen para poder realizar minería de datos a la información.

Independientemente de la ubicación dentro de la clasificación al problema por solucionar o de la técnica empleada, la minería de datos trabaja para encontrar información de valor dentro de los grandes datos en un proceso automatizado para ayudar en la toma de decisiones.

La minería de datos hace parte de un proceso más grande llamado Descubrimiento de Conocimientos en Base de Datos (KDD – *Knowledge Discovery in Databalese*), es un proceso de extracción de conocimiento. La minería de datos se restringe únicamente a la obtención de modelos.

La figura 13 muestra el diagrama del proceso KDD, es importante notar que dentro del citado proceso la minería de datos es el penúltimo paso (previsión de modelos en minería de datos, en donde inicialmente se tienen los grandes datos de distintos medios, luego se seleccionan algunos mediante consultas, se procesan y transforman pensando en etapas siguientes, para luego pasar a la etapa de minería de datos en la que se establecen modelos y patrones, para que finalmente sean interpretados para presentarlos como el conocimiento obtenido.

Figura 13. Proceso KDD. Fuente L. Vieira Braga, L. Ortiz Valencia, S. Ramírez Carvajal 2009.

En minería de datos se destacan dos metodologías, CRISP-DM (Cross Industry Standard Process for Data Mining), la cual es utilizada por Clementine – SPSS y SEMMA (Sample, Explore, Modify, Model, Access) empleada por SAS Enterprise Miner. Ambos sistemas pasan por las mismas etapas: recoger datos, depurarlos y analizarlos, para luego construir un modelo predictivo (con diferencias en las etapas de presentación e implementación).

Genéricamente un proyecto de minería de datos se realiza en las siguientes etapas:

- Definición del problema: conocer perfectamente lo que el estudio desea obtener, en común con todos los involucrados en el proyecto.
- Adquisición y evaluación de datos: adquirir, formatear y validar los datos, tomar muestras aleatorias.
- Extracción de características: identificar atributos que contribuyan en la solución, los que no se alteran deben salir del estudio, para producir un *dataset* representativo y confiable.
- Plan para el prototipo: desarrollo de hipótesis y del prototipo
- Desarrollo del modelo: desarrollar modelos descriptivos y/o predictivos.
- Evaluación del modelo: considerar los resultados del prototipo.
- Implementación: presentación del producto final.
- Evaluación del retorno de la inversión: se evalúa si la inversión en el proyecto está generando utilidades para los inversionistas.

3.2 Minería de opinión

La minería de opinión también se conoce como, *análisis de opiniones, análisis de sentimientos, extracción de opinión, minería de sentimiento, análisis de subjetividad, análisis de*

emociones, o en inglés sentiment analysis u opinion mining sin que haya diferencia entre estos términos.

La minería de opinión o análisis de sentimientos se ha estado estudiando a partir del año 2000. Se ha abordado con técnicas de aprendizaje automático (Pang, Lee y Vaithyanathan, 2002) o no supervisado (Turney, 2002). Con estos trabajos pioneros en el área se determinó la polaridad de opiniones, aplicándolas a las opiniones sobre películas, posteriormente, por medio de técnicas combinadas se han realizado estudios en el mismo tema y desde luego en otros dominios (Pla Ferran y Hustado Lluis, 2013).

El análisis de sentimientos surge de la necesidad de clasificar la orientación o la opinión de lo manifestado en un documento, en el presente proyecto examinar la subjetividad del documento es relevante, y aplicar técnicas de clasificación no son suficientes.

El análisis de sentimientos (AS) clasifica los documentos en función de la polaridad de la opinión que se expresa, identificando las opiniones positivas, negativas y neutras. Se utiliza para determinar la polaridad de las opiniones del documento o frase, y en determinar si el documento contiene opiniones. El análisis se realiza aplicando *aprendizaje automático* o por medio del *enfoque semántico*.

El AS ha sido aplicado a muchos dominios, la mayoría de ellos para documentos en inglés, siendo muy reducido este tipo de investigaciones en idioma español. Las universidades españolas, Universidad de Sevilla y su grupo ITALICA, y la Universidad de Jaén con el grupo SINAI (Sistemas Inteligentes de Acceso a la Información) han realizado sendos trabajos, sin embargo ésta última realizando el AS en lengua Árabe también.

3.3 Microblogging

Es un servicio que permite a sus usuarios enviar y publicar mensajes cortos de texto generalmente de hasta 140 caracteres, la comunicación fluye por medio de sitios WEB, mensajes SMS, mensajería instantánea o por medio de aplicaciones específicas para tal fin las cuales pueden ser desarrollados por terceros, inclusive para dispositivos móviles. Tiene como objetivo permitir compartir información con otros usuarios e interactuar con ellos por medio de mensajes y enlaces a otras publicaciones WEB. Tiene cuatro componentes:

- Es un Blog ya que el autor publica contenidos en orden cronológico.
- Posee un sistema de mensajería instantánea, ya que permite conversaciones en tiempo real.
- Permite enviar mensajes de texto SMS o similares de hasta 140 caracteres, hoy en día Twitter permite mensajes de hasta 280 caracteres.
- Es una red social porque los mensajes del usuario pueden ser leídos por sus seguidores para interactuar con ellos y compartir estados o mensajes entre pares de la comunidad social.

3.4 Twitter

Twitter es una red social clasificada como un servicio de microblogging, su número de usuarios (aproximadamente 500 millones de usuarios de los cuales 360 siguen activos en Enero de 2016) y de mensajes continúan en aumento (65 millones de tweets al día (Twitter, 2016) [23]), aunque para algunos analistas empieza a ser lento, como lo cita CNN [24] en español (en su artículo *"Desastre en Twitter: pierde millones de usuarios y sus acciones se hunden"*) quienes

informan que la red social cuenta con 305 millones de usuarios activos (una quinta parte de Facebook) luego de haber perdido 2 millones de usuarios en el último trimestre de 2015; la cantidad de usuarios activos y el amplio uso que se ha dado a éste medio social y de comunicación, la mantienen como una de las favoritas, además de haber tenido crecimientos importantes en años anteriores. Es una empresa que factura 2.500 millones de dólares anuales y tiene un valor en bolsa de valores de 10.000 millones de dólares.

Twitter fue creada por Jack Dorsey en 2006 en California, su sede principal se encuentra en San Francisco; se trata de un servicio de microblogging que permite enviar mensajes de texto plano cortos (conocidos como tweets) hasta de 280 caracteres, los cuales se muestran en el perfil del usuario (por defecto los mensajes son públicos). En Noviembre de 2009 aparece la versión de Twitter en español. Los usuarios de Twitter pueden suscribirse (se conoce como *seguir*) a otros perfiles de usuario del sistema, además los usuarios pueden tener "seguidores", los seguidores o "followeres" son los usuarios que siguen una cuenta o perfil en el sistema Twitter. Los usuarios pueden enviar mensajes directos iniciando su mensaje con @ seguido del nombre de usuario de destino, para lo cual los usuarios deben seguirse mutuamente, además es posible agrupar los mensajes de un determinado tema utilizando el símbolo "#" almohadilla al que se le conoce como *hashtag.*

Twitter permite acceder de manera WEB al sistema, sin embargo es desde aplicaciones externas para Smartphone en las principales plataformas desde donde se realiza la mayor parte de las comunicaciones, también vía mensajes SMS con cargos según el operador móvil del país.

Twitter está escrita en el framework Ruby on Rails (RoR) y en el lenguaje de programación Scala; para la interfaz WEB y para programar el servidor que almacena los mensajes,

respectivamente. Dispone de una API abierta a todo tipo de desarrolladores en aplicaciones WEB de escritorio o para móviles.

Por medio de la API llega más del 50% del tráfico a Twitter. Las aplicaciones de terceros deben usar obligatoriamente el protocolo OAuth (Open Authorization) el cual permite autorización segura de la API en aplicaciones de escritorio, WEB y móviles. Para Twitter éste tipo de autenticación aumenta la seguridad y mejora la experiencia.

3.5 Análisis de sentimientos en twitter

A partir del 2009 se cuenta con referencia de trabajos importantes en los que se aplica Análisis de sentimientos (AS) a Twitter, con la característica que es para idioma Inglés, como lo señala Eugenio Martínez Cámara en su *artículo "Análisis de Sentimientos"* citando por ejemplo los siguientes trabajos: *"Twitter Sentiment Classication using Distant Supervision"* en 2009, *"From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series*" en 2010 y *"Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment"* en 2010.

En agosto de 2009 la empresa de investigaciones de mercado *Pear Analytics* realizó un estudio con 2.000 tweets enviados desde Estados Unidos (en un periodo de dos semanas entre las 11:00 y 17:00 horas), clasificándolos en seis categorías distintas (figura 14):

- Charlas sin sentido (40%)
- Conversaciones (38%)
- Retweets (9%)
- Autopromoción (5%)

- Mensajes SPAM (4%)
- Noticias (4%)

Danah Boyd, quien es investigadora en redes sociales, considera que las *charlas sin sentido* podrían encajar dentro de lo que se conoce como “sensibilización periférica” o “acicalamiento social”, el cual consiste en querer saber que piensan, hacen y sienten las personas a su alrededor [25].

A pesar de que éste estudio fue realizado hace siete años y apenas toma una muestra de dos mil tweets, deja ver una porción amplia de lo que comunican los usuarios en sus mensajes, algo importante en el presente estudio porque aproximadamente en un 80% de los mensajes, los usuarios comunican sus sentimientos, lo que pasa a su alrededor o lo que les está ocurriendo; lo anterior podría mostrar características en común para el seguimiento a la actividad de un grupo de personas en un espacio y tiempo determinado, como lo es por ejemplo un encuentro deportivo previamente programado y de amplio conocimiento en la sociedad.

En Colombia, la edad promedio de los usuarios de Twitter (twiteros) es de 24 años; en el mundo el 80% tienen menos de 29 años, el 43% tienen menos de 19 años y tan solo el 7% tiene más de 40 años. Cifras que concuerdan con los estudios que muestran la participación de los jóvenes dentro de las barras bravas, además son los jóvenes menores de edad los que generalmente inician los problemas escudándose en su edad para no tener problemas judiciales en Colombia (Colombia.com en 2002).

En 2014, Twitter en Colombia tenía 17 millones de cuentas registradas de las cuales 4,4 millones (26%) pertenecían a usuarios activos.

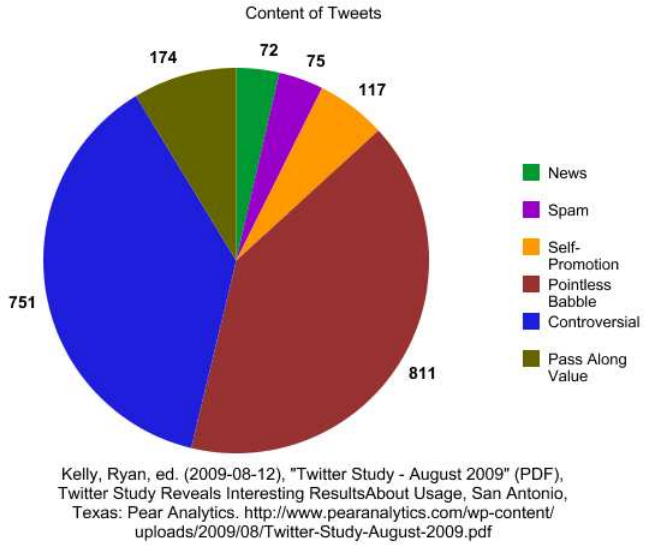


Figura 14. Contenido de los tweets por Pear Analytics en 2009 [23].

Según la clasificación del motor de búsqueda Alexa.com (compañía filial de amazon.com) en Junio de 2016, twitter.com fue el décimo segundo (12°) sitio WEB más visitado en Colombia (8° a nivel mundial), superando a Instagram.com (14°) y a Whatsapp.com (21°). Youtube.com ocupó el primer lugar seguido por Google.com.co y Facebook.com respectivamente (fuente el autor).

3.6 Facebook

Facebook es un sitio WEB bajo la modalidad de *red social* que recrea una comunidad virtual en la que sus integrantes comparten gustos, sentimientos, fotos, conversaciones por chat, videos, línea de tiempo o biografía (anteriormente muro) y enlaces. En sus inicios fue un sitio WEB para los estudiantes de la Universidad de Harvard, diseñado para que tuvieran comunicación fluida, compartiendo contenido con sencillez por medio de Internet, luego en 2006 estuvo disponible para cualquier usuario de la WEB.

Fue creada por Mark Zuckerberg, un estudiante de la Universidad de Harvard quien abandonó sus estudios luego de tener problemas con directivas de la Universidad quienes lo inculparon de uso indebido de información de los estudiantes, con su primer sistema de red social, muy pronto y luego de asociaciones problemáticas con otras personas, su red social había ganado mucha popularidad, al punto de ser utilizada por otros estudiantes en distintas universidades de Estados Unidos. TheFacebook fue el nombre con el que inicio el proyecto, desde 2004 Facebook es el nombre oficial. Cuenta en 2016 con 1650 millones de usuarios activos mensuales, de los cuales el 50% se conectan con la plataforma diariamente, de ellos más de 700 millones se conectan por medio de móviles.

Son distintas las razones por las cuales Facebook ha crecido tan rápido en todas sus dimensiones, como lo es la innovación, popularidad, alianzas estratégicas, entre otras, sin embargo ha sido abrir la plataforma a terceros para que estos desarrollen aplicaciones lo que ha permitido crecer y tener variedad de aplicaciones asociadas, que van desde aplicaciones para conectarse al sistema desde móviles, hasta juegos ampliamente conocidos como Candy Crush, Farm Ville, Dragon City, Criminal Case entre muchos otros. La figura 15 muestra la popularidad de la red social a partir de 2004 hasta 2015, donde se observa el vertiginoso crecimiento a partir de 2008.

Por su parte, un poco más del 50% de los usuarios de Facebook se encuentran entre los 18 y 34 años, seguidos del segmento de personas entre los 13 y 17 años con un 20%, es decir que por lo menos el 70% de los usuarios de Facebook se encuentran entre los 13 y 34 años de edad. La figura 16 muestra los rangos de edad y los respectivos porcentajes, información suministrada en 2011.

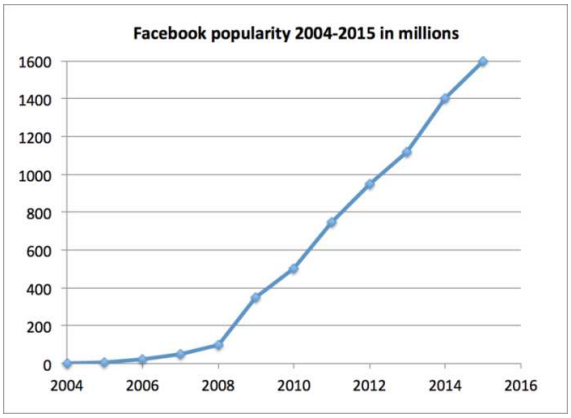


Figura 15. Usuarios de Facebook entre 2004 y 20015. Fuente Wikipedia

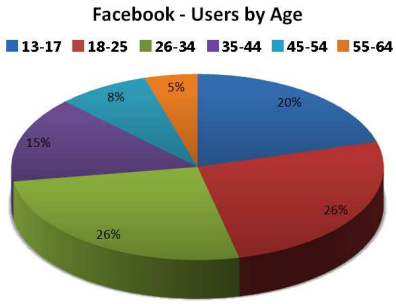


Figura 16. Porcentaje de usuario en Facebook por edad. Fuente Wikipedia.

Al igual que Twitter, Facebook fue traducida al español por usuarios voluntarios de manera no remunerada, en 2007 se lanzaron las versiones en alemán, francés y español.

Colombia es uno de los 10 países de habla hispana con más usuarios en Facebook (En 2013 llegó a 12.675.000 usuarios), la acompaña Perú, Ecuador, Venezuela, Chile, y Argentina. La figura 17 señala los países, información obtenida en 2011.

Figura 17. Países hispanos con más usuarios en Facebook. Fuente Wikipedia.

En cuanto a su infraestructura, ésta se encuentra en grandes centros de datos alquilados en Estados Unidos, con una red de más de 50.000 servidores en 2010, con sistemas operativos GNU/Linux usando la combinación de tecnologías LAMP en su servidor WEB.

3.7 R

R es un entorno de lenguaje de programación de código abierto (software libre), es diseñado para computación y generación de gráficas estadísticas por medio de carga de bibliotecas o paquetes, lo que la hace una herramienta idónea para el análisis estadístico de grandes conjuntos y cantidades de datos, minería de datos, investigación biométrica, bioinformática y matemáticas financieras. Fue creado en la Universidad de Auckland de Nueva Zelanda en 1.993 por Ross Ihaka y Robert Gentlemean del departamento de estadística, es software libre, un proyecto GPL/GNU. R es el resultado de la combinación de las fortalezas de sus predecesores lenguajes de programación para estadística S (de donde toma su apariencia) y Sheme (de donde obtiene la semántica). Se encuentra disponible para plataformas UNIX/LINUX, MAC y Windows.

R brinda una amplia variedad de estadísticas (modelos lineales y no lineales, estadística clásica, análisis de series temporales, clasificación y agrupación) y técnicas de graficación.

R permite a sus usuarios crear sus propias funciones, permite integrarse a bases de datos. Por medio de bibliotecas puede ser utilizado desde Perl y Python, su poder se compara con GNU Octave y MATLAB. Cuenta con una interfaz para interactuar con Weka (RWeka) y enriquecer el lenguaje R con algoritmos de minería de datos.

Figura 18. Logo. Fuente r-project.org

En cuanto a las conexiones de R con Twitter y Facebook, se cuenta con APIs para obtener servicios de estas redes sociales de forma sencilla, son ellas TwitteR y Rfacebook respectivamente. Además cuenta con más de 10 interfaces gráficas e IDEs como Sage, JGR, RStudio, Eclipse, Emacs, Vim, Scite, notepad++, entre otros. Su última versión es la 3.4.3, disponible desde el 30 de abril de 2017 en el sitio oficial https://www.r-project.org/. El logo oficial de R en 2016 aparece en la figura 18 con licencia bajo los términos de Creative Commons Attribution-ShareAlike 4.0 International license (CC-BY-SA 4.0).

Capítulo 4
Implementación

4.1 Diagrama de bloques para el desarrollo del proyecto

Para el desarrollo del problema de estudio, se plantea el siguiente modelo de proceso (ver la figura 19), el cual se encuentra en etapas. Cada una de las etapas requiere que culmine la anterior. Este modelo de proceso es una particularización muy aproximada al modelo KDD, citado anteriormente en el presente documento y por muchos autores. La etapa de filtrado se realiza mediante estadística con el lenguaje de programación R.

Figura 19. Modelo de bloques del proceso en el proyecto.

El presente proyecto (Seguimiento a las barras bravas del futbol en la ciudad de Pereira, basado en tecnologías big data) propone un modelo de seguimiento y verificación al comportamiento de las barras bravas de futbol en Pereira, se soporta en tecnologías Big Data porque grupos de personas con intereses en común con más frecuencia publican por redes sociales sus actividades, opiniones, comentarios. En redes sociales como Twitter estos volúmenes de datos son administrados como grandes datos.

La figura 20 muestra el desarrollo del modelo de bloques de forma técnica y ampliada.

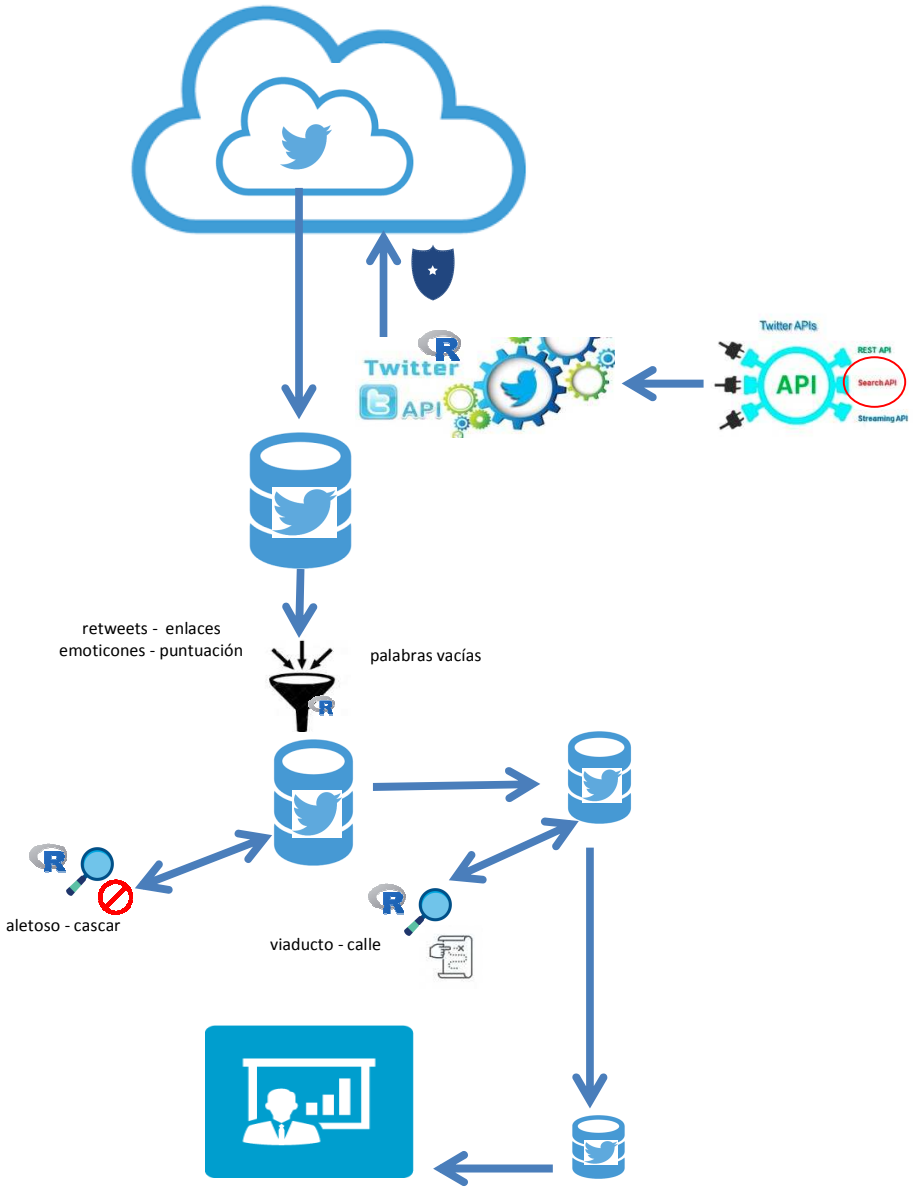


Figura 20. Desarrollo del modelo de bloques del proceso en el proyecto.

Cada una de las etapas del modelo se amplía a continuación:

4.1.1 Recolección de datos: en ésta etapa inicial del modelo, se determinan las fuentes de datos y el tipo de información que se va a utilizar. Es en ésta etapa donde se toman los datos relevantes para el análisis desde la fuente de datos.

En esta etapa inicial del modelo se almacena la información del gran conjunto de datos de la red social Twitter. Tales volúmenes de datos son mantenidos y administrados por la respectiva red social, es decir, todos los detalles tecnológicos como servidores, copias de respaldo, interfaces o APIs con el usuario, entre otros detalles están a cargo de los administradores TI de las redes sociales. Los usuarios de las redes sociales por medio de aplicaciones WEB propias de las redes sociales o por las desarrolladas por terceros, acceden a sus respectivos perfiles desde los cuales generan nuevos contenidos (estados, mensajes) y gestionan su propia información.

La conexión con estos datos se realiza por medio del paquete (API) TwitteR, a través de consultas y servicios streaming utilizando el lenguaje de programación R en etapas siguientes al modelo.

4.1.2 Almacenamiento: en ésta etapa del modelo se realizan tres operaciones (extracción, transformación y carga), como se había mencionado con anterioridad, cada etapa es insumo para la siguiente. El resultado de ésta etapa es el conjunto de datos (dataset) obtenido de la red social.

- **Extracción**: etapa que permite crear conjuntos de datos específicos, para no traer cantidades de datos como los que administra la red social objetivo. Se realiza por medio de conexiones utilizando la API twitteR, por periodos de tiempo (o por defecto los últimos diez días), idioma de los mensajes (en éste caso español); la ubicación geográfica (desactivada en las consultas de éste estudio debido al escaso número de mensajes con

geo referenciación), acordes a la fecha y ubicación de los encuentros deportivos. Los atributos que permanezcan estáticos o no cambien no son tenidos en cuenta. En ésta etapa, las consultas se realizan mediante criterios específicos relacionados con el Dominio de la Investigación, los cuales son los nombres con los que se conoce al Deportivo Pereira y algunos de los equipos de futbol que tienen barras de futbol reconocidas en el área Metropolitana. La extracción puede ser a las líneas de tiempo de usuarios específicos, exponiendo una consulta al flujo de datos actual de la red social y consultando la base de datos en un rango de tiempo determinado.

- **Transformación**: en esta sub etapa del modelo, no tenemos interacción directa con las respectivas redes sociales, el proceso de extracción suministra los datos, los cuales son transformados de lista de tweets a tuplas para ser almacenados en una matriz más general de datos llamada técnicamente como Data Frame, cada tupla se compone de 16 campos, todos suministrados por la red social.
- **Carga:** esta sub etapa permite crear el conjunto de datos de estudio o Data Set objetivo en un archivo externo tipo CSV (comma - separated values por su sigla en inglés, es decir valores separados por comas), los cuales permiten mejorar el tiempo de carga de la información en comparación con archivos similares almacenados en hojas de cálculo como Microsoft Office Excel.

4.1.3 Técnicas de filtrado: en ésta etapa del modelo se filtran los mensajes de texto cargados, denominados como Data Set, para que la computadora los pueda entender y analizar en etapas siguientes. Se eliminan de los mensajes los enlaces, retweets, menciones, palabras vacías, ya que ésta información no aporta mayores detalles al proceso.

4.1.4 Análisis: en esta etapa usando técnicas estadísticas en el lenguaje de programación R, se seleccionan aquellos mensajes que reúnen características de comportamientos inapropiados, promoción o aliento a contravenciones como riñas callejeras, hurtos, entre otros, como también reuniones en lugares públicos o convocatorias masivas de personas que pueden afectar la convivencia y seguridad ciudadana.

4.1.5 Conocimiento: ésta etapa concluye con la propuesta del modelo. Es la última etapa y es la encargada de los mensajes previamente cargados y que deben de generar una advertencia dentro del seguimiento a los grupos de barras bravas (barras activas) de la ciudad de Pereira.

4.2 Ejecución del proyecto

El desarrollo procedimental del proyecto inicia con la recolección de datos generados por los usuarios de la red social Twitter, para su posterior análisis.

Twitter cuenta con por lo menos tres APIs (Application Programming Interface) en el lenguaje de programación R con las cuales se pueden obtener flujos de datos. Debido a la cantidad de mensajes que los usuarios de la red social generan por segundo, la que a su vez es almacenada como grandes datos en sus centros de almacenamiento, se requiere que al momento de obtener información se tengan consideraciones precisas para extraer subconjuntos de datos relevantes para la investigación. Se deben considerar: la cadena de búsqueda, lenguaje, rangos de fechas, ubicación geográfica aproximada y tipo de conexión para realizar la consulta. A continuación el listado de los tipos de conexión con que cuenta la API para la interacción con Twitter:

- **Streaming API:** proporciona un flujo continuo de *tweets* para conformar un subconjunto de datos en formato *json* casi en *tiempo real* de las publicaciones de los usuarios de

Twitter, por medio de una conexión *http* permanente en el tiempo establecido. La velocidad de recepción de los tweets es dependiente del ancho de banda de los extremos en las conexiones y de la sobre carga en los servidores de Twitter.

- **SearchTwitter API:** suministra los tweets publicados en los últimos siete días de acuerdo a la consulta realizada, hasta donde puede recuperar los últimos 8000 mensajes aproximadamente, entrega el subconjunto de datos en formato *json o atom.* Permite aplicar filtros por lenguaje y localización. Esta limitada hasta 150 peticiones por hora, por usuario o por IP.
- **REST API:** permite consultar en toda la base de datos disponible en Twitter, de la que puede recuperar los últimos 3200 tweets. Soporta formatos json, xml, rss, atom. Al igual que Search API, está limitada hasta 150 peticiones por hora, por usuario o por IP.

4.2.1 Recolección de datos

Twitter es un servicio de Microblogging con por lo menos 500 millones de usuarios que generan 65 millones de mensajes o tweets al día, maneja más de 800.000 peticiones de búsqueda por día. Desde finales del mes de septiembre de 2017 la longitud de sus mensajes ha pasado de 140 caracteres a 280, cambio que no se realizó a sus versiones en coreano, japonés y chino, ya que a base de símbolos logran comprimir mensajes y comunicar sus ideas, caso contrario a otros idiomas en donde las críticas por su restricción de apenas 140 caracteres han dado pie para que la red social tome un segundo aire e inicie con la era la facilidad de expresión al duplicar la capacidad de sus mensajes.

Como se ha mencionado el sistema Twitter permite enviar mensajes de texto plano de corta longitud, con máximo 280 caracteres, desde la interfaz web oficial (https://twitter.com/) o desde

aplicaciones desarrollados por terceros para acceder a la plataforma, en ambos casos se requiere previo registro en la red social.

La interfaz web se encuentra escrita en Ruby on Rails (RoR) el cual es un Framework de código abierto para aplicaciones web escrito en Ruby que sigue el paradigma Modelo vista controlador (MVC). En cuanto a los mensajes se encuentran almacenados en un servidor programado en Scala, el cual permite expresar patrones comunes de programación integrando características de lenguajes funcionales y orientados a objetos.

La interfaz web dispone de una API abierta a terceros, según Biz Stone – *cofundador de Twitter*, más del 50% del tráfico llega por medio de la API que permite la programación de aplicaciones para móviles o equipos de escritorio.

4.2.2 Almacenamiento

En ésta etapa se realizan las operaciones de extracción, transformación y carga (ver figura 21). El resultado de ésta etapa es el conjunto de datos (dataset) objeto para la investigación.

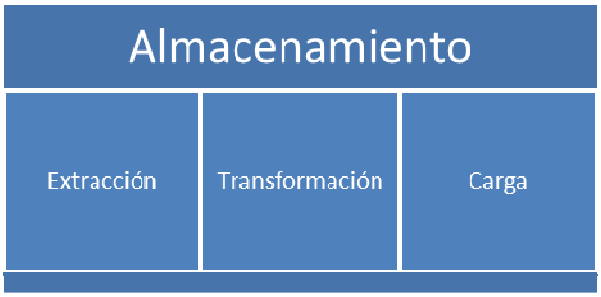


Figura 21. Etapa de almacenamiento.

Extracción: en esta sub etapa las consultas se realizan mediante la combinación de expresiones y operadores lógicos, sobre líneas de tiempo a usuarios específicos o exponiendo una consulta al flujo de datos en Twitter y al que se está generando en un momento determinado.

Se define una función para la extracción de datos, con dominio dentro del conjunto de mensajes de la red social, y rango o Dataset inicial para el análisis del modelo, como se representa en la figura 22.

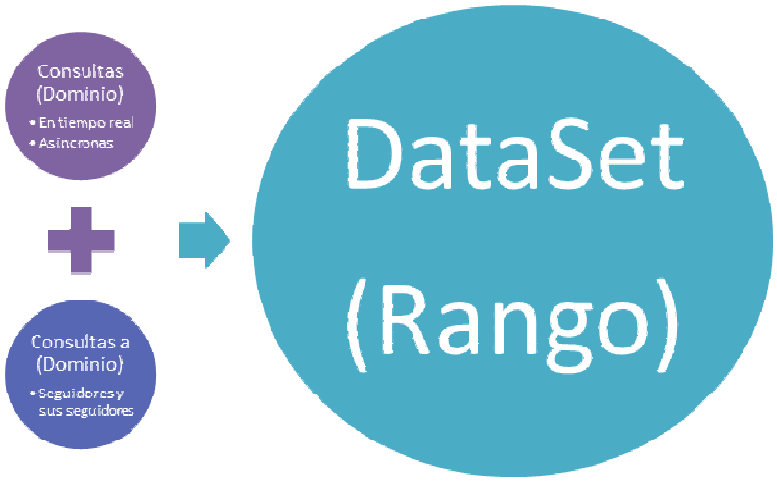


Figura 22. Dominio y rango en el proceso de extracción de datos.

La API TwitteR desarrollada para el lenguaje de programación R, permite realizar conexiones a Twitter para la extracción de información de tres maneras distintas:

- Mensajes en tiempo real
- Mensajes en la línea de tiempo de un usuario determinado
- *Mensajes de los últimos 10 días de todos los perfiles*

En la figura 23 se presentan las tres estrategias para la extracción de información a Twitter y las respectivas funciones en el lenguaje de programación R.

Figura 23. Estrategias de extracción de información con la API TwitteR

Por ejemplo para consultar a Twitter 500 mensajes que incluyan las palabras *Pereira y América* dentro de los mensajes en español, se escribe una sentencia como la siguiente:

tweets <- **searchTwitter**("pereira && america", n = 500, lang = 'es')

O emplear el operador OR "||" para aplicar un filtro más amplio, como sigue:

tweets <- searchTwitter("pereira || america", n = 500, lang = 'es')

Tweets en tiempo real

Para realizar captura de mensajes (tweets) streaming en tiempo real, se emplean las librerías StreamR y ROAuth, principalmente, sin embargo otras librerías también son utilizadas, como los son:

- ✓ *library(bitops)*
- ✓ *library(RCurl)*
- ✓ *library(rjson)*

- ✓ *library(twitteR)*
- ✓ *library(tm)*
- ✓ *library(RColorBrewer)*
- ✓ *library(wordcloud)*
- ✓ *library(ggmap)*
- ✓ *library(Rcpp)*
- ✓ *library(stringr)*

En la siguiente línea de código se crea un archivo en json con la muestra de 100 tweets con la coincidencia de la palabra “pereira” al momento de ser lanzado en la consola de RStudio. Ésta consulta está se expone o activa durante 30 segundos como se observa en el parámetro timeout, y se detiene antes de tiempo si el número de mensajes logra ser completado antes del tiempo de exposición de la captura.

filterStream*(file.name="tweets_consulta.json", track="pereira", tweets=100, oauth=autorizacion, timeout=30, lang="es")*

Luego se mapea el resultado de la consulta en la variable consulta. La variable consulta contiene en forma de data frame toda la información de manera estructurada, ver la siguiente línea de código.

consulta <- parseTweets(tweets='tweets_*consulta*.json')

Con el comando *View(consulta)* se observar el contenido del *dataframe.*

El script debe tener la ejecución del comando OAuthFactory$new y autorizacion$handshake

A continuación un ejemplo del script:

```
# Se crea el certificado solo una vez
download.file(url="http://curl.haxx.se/ca/cacert.pem",destfile="cacert.pem")

consumer_key <- 'xxxxxxxxxxx'
consumer_secret <- 'yyyyyyyyyy
access_token <- 'zzzzzzzzzzzzzzz
access_secret <- 'wwwwwwwwwww'

autorizacion <- OAuthFactory$new(consumerKey=consumer_key,
consumerSecret=consumer_secret,requestURL='https://api.twitter.com/oauth/request_token',acce
ssURL='https://api.twitter.com/oauth/access_token',authURL='https://api.twitter.com/oauth/auth
orize')
autorizacion$handshake(cainfo='cacert.pem')
```

Al ejecutar el comando ***filterStream*** y si hay coincidencias en el filtro programado, se obtienen registros con 42 columnas con los siguientes campos:

1. Tweet o mensaje (text)
2. Contador de retweets (retweet_count)
3. Favorito
4. Truncado
5. Identificador del mesaje (id_str)
6. in_reply_to_screen_name
7. Origen del mensaje (source)
8. Retweeted

9. Fecha de creación del mensaje (created_at)
10. in_reply_to_status_id_str
11. in_reply_to_user_id_str
12. Lenguaje (lang)
13. listed_count
14. verified
15. location
16. Identificador de usuario (user_id_str)
17. Descripción del perfil (description)
18. Georefenciación habilitada (geo_enabled)
19. Fecha de creación del perfil (user_created_at)
20. statuses_count
21. Número de seguidores (followers_count)
22. favourites_count
23. protected
24. Página Web del perfil (user_url)
25. Nombre del perfil (name)
26. Zona horaria (time_zone)
27. Lenguaje del usuario (user_lang)
28. utc_offset
29. Perfiles seguidos por el usuario (friends_count)
30. Nombre de usuario en Twitter (screen_name)
31. Código del país (country_code)

32. País (country)
33. place_type
34. full_name
35. place_name
36. place_id
37. Coordenada latitud del mensaje (place_lat)
38. Coordenada longitud del mensaje (place_lon)
39. Lat
40. Lon
41. expanded_url
42. url

Tweets hasta diez días atrás

Para capturar mensajes (tweets) almacenados en la base de datos de Twitter hasta con diez días de anterioridad, se emplean las mismas librerías que en el caso *Tweets en tiempo real.*

En la siguiente línea de código se presenta la manera de acceder a los últimos 100 tweets con la coincidencia “pereira” dentro de los últimos diez días:

tweets1 <- searchTwitter("pereira", n = 100)

Posteriormente se convierten los datos tweets1 en una data frame llamado *tweets1.df*

tweets1.df <- do.call("rbind", lapply(tweets1, as.data.frame))

Con el comando *View(tweets1.df) se* puede observar el resultado en el *dataframe.*

Previamente es necesario crear la aplicación en la plataforma Twitter para obtener las credenciales de autorización.

El script debe tener la ejecución del *comando setup_twitter_oauth* la cual incluye los campos *consumer_key, consumer_secret, access_token y access_secret.*

A continuación un ejemplo del script:

consumer_key <- 'xxxxxxxxxxx'

consumer_secret <- 'yyyyyyyyyy

access_token <- 'zzzzzzzzzzzzzzzz

access_secret <- 'wwwwwwwwwww'

setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)

Tweets en toda la línea de tiempo de Twitter

Para recopilar los mensajes de un perfil determinado se utiliza la función userTimeline, permite recuperar los últimos 3200 mensajes.

En la siguiente línea de código se presenta la manera de recopilar a los últimos 1000 tweets del perfil “superdepor”:

tweets = userTimeline(“superdepor”, 1000) # Lo hacemos para los últimos 1000 tweets

Transformación: en ésta sub etapa, por medio del lenguaje de programación R se crea un data frame para convertir los mensajes recolectados desde Twitter. Un data frame es una estructura de datos similar a una matriz, ya que es una matriz de datos más generalizada que las matrices convencionales porque permiten dentro de sus celdas tener diferentes tipos de datos, sin embargo los elementos de cada columna deben tener el mismo tipo de dato.

Carga: ésta sub etapa nos permite cargar ciertas variables del data frame, en total se cuenta con 16 columnas. Cargar nuestro dataset en forma de dataframe permitirá la realización de las etapas siguientes. Se recomienda en futuras investigaciones, almacenar la información en una base de datos No-SQL.

4.2.3 Técnicas de filtrado: en ésta etapa, se parte del conjunto de datos (dataset) almacenado, inicialmente extraídos de la red social, para eliminar de los mensajes enlaces, retweets, menciones y palabras vacías. En el presente proyecto a partir de un data frame se aplican filtros para el posterior procesamiento de lenguaje natural.

Por ejemplo:

```
# Remover retweets

txtclean = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", txt)

# Remover menciones a otras cuentas @cuenta

txtclean = gsub("@\\w+", "", txtclean)

# Remover símbolos de puntuación

txtclean = gsub("[[:punct:]]", "", txtclean)

# Remove números

txtclean = gsub("[[:digit:]]", "", txtclean)

# Remover links/enlaces

txtclean = gsub("http\\w+", "", txtclean)
```

Se recomienda realizar un filtrado preliminar a partir del mensaje original, pasando todas las palabras a minúsculas, suprimiendo los dobles espacios, signos de puntuación y caracteres que forman expresiones como la carita feliz (dos puntos y paréntesis); luego se eliminan las palabras vacías (pronombres, conectores) ya que no aportan información. Posteriormente se convierten todos los verbos a infinitivo o a su raíz. Finalmente se crea un listado de las palabras con la frecuencia de cada de ellas, éste procedimiento de repite con cada uno de los mensajes. El procedimiento se representa en la figura 24.

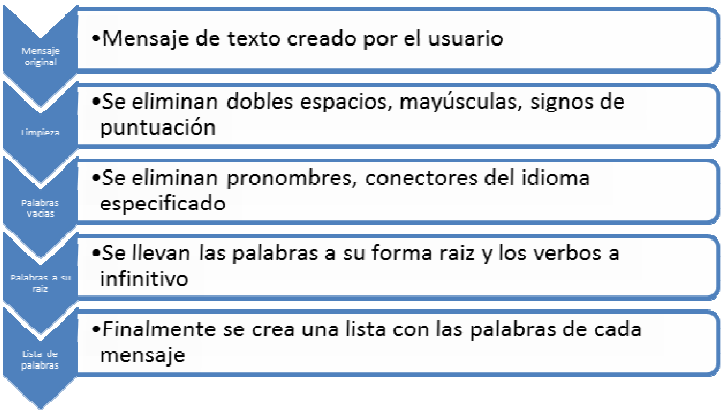


Figura 24. Procedimiento de filtrado de mensajes.

4.2.4 Análisis

En esta etapa se realiza un filtrado de los mensajes bajo el criterio del listado de palabras o vocabulario que dentro del dominio del problema tienen especial significado, este *lexicón* se aplica al Data Set. Es importante destacar que con este análisis realizado por medio de filtros se hace complicado entender ciertos mensajes, ya que dentro del lenguaje natural se presenta ambigüedad con las palabras, homónimos, metáforas y sarcasmos, así como errores ortográficos, palabras recortadas, signos de puntuación con significados, entre otros.

El presente proyecto busca patrones coincidentes para identificar comentarios amenazantes, ofensivos o sospechosos, determinar temas de interés, encontrar los actores generadores de opinión en lo relacionado con las barras bravas o recientemente llamadas barras activas que puedan o afecten la convivencia y seguridad ciudadana.

4.2.5 Conocimiento

Es en esta etapa en la que se entrega al analista humano un consolidado de información con los mensajes (tweets) con contenido que muestra que se ha afectado o que se puede afectar la convivencia y seguridad ciudadana.

En esta fase se puede aplicar a la base de conocimiento un lexicón con algunos lugares de la ciudad en el que han ocurrido incidentes de convivencia y seguridad ciudadana derivados de los encuentros deportivos y del comportamiento de sus hinchas.

4.3 Códigos en R

Esta sección muestra el desarrollo de los procesos descritos en la Implementación, tomando en el estudio a la red social Twitter. Los scripts son desarrollados en el lenguaje de programación R, realizan los siguientes procesos:

1. Presentación del software empleado.
2. Instalación y carga de librerías.
3. Autenticación a Twitter.
4. Almacenamiento
5. Técnicas de filtrado.

6. Análisis con R.

7. Conocimiento.

4.3.1 Presentación del software empleado

El desarrollo de la presente implementación se ha realizado utilizando el lenguaje de programación *R* sobre el sistema operativo Windows. *R* se utiliza para la estadística computacional y la graficación. Puede ser descargado de la página oficial del proyecto: https://www.r-project.org. Luego de la descarga e instalación del lenguaje de programación *R* se debe instalar *RStudio* en su versión para escritorio; el cual es el Entorno de Desarrollo Integrado (IDE) para el lenguaje de programación *R*, se encuentra disponible para la descarga en: www.rstudio.com/products/RStudio/

4.3.2 Instalación y carga de librerías

Una vez instalado el lenguaje de programación y el entorno de desarrollo para él se debe proceder con la instalación de las librerías, es un procedimiento que solo se realiza una vez dentro de la preparación del sistema. Para instalar una librería se debe lanzar desde la línea de comandos dentro de paréntesis en comillas dobles anteponiendo el comando *install.packages.* Es indispensable tener conectividad a internet para obtener el paquete. Para el desarrollo del proyecto se han utilizado los siguientes paquetes, a continuación con su comando de instalación:

```
Source on Save
 1  install.packages("bitops")
 2  install.packages("igraph")
 3  install.packages("RCurl")
 4  install.packages("rjson")
 5  install.packages("twitteR")
 6  install.packages("tm")
 7  install.packages("RColorBrewer")
 8  install.packages("wordcloud")
 9  install.packages("ggmap")
10  install.packages("Rcpp")
11  install.packages("stringr")
12  install.packages("XML")
13  install.packages("xlsx")
14  install.packages("rJava")
15  install.packages("lattice")
16  install.packages("ROAuth")
17  install.packages("streamR")
18  install.packages("caret")
19  install.packages("httr")
20  install.packages("SnowballC")
21  install.packages("RJavaTools")
22  install.packages("openxlsx")
23
```

Como se ha mencionado, los paquetes se instalan una única vez desde la línea de comandos, para lo que se debe tener conectividad con internet para obtener los respectivos paquetes solicitados. Luego y cada vez que se inicie el sistema *R y RStudio* es necesario llamar las librerías correspondientes a los respectivos paquetes instalados, se realiza desde la línea de comandos con el comando *library* encerrando en paréntesis el nombre de la librería, como se muestra a continuación:

```
Source on Save
 1  library(bitops)
 2  library(RCurl)
 3  library(rjson)
 4  library(twitteR)
 5  library(igraph)
 6  # libreria Tex Mining para Mineria de textos
 7  library(tm)
 8  library(RColorBrewer)
 9  # libreria Word Cloud para Mineria de datos
10  library(wordcloud)
11  library(ggmap)
12  library(Rcpp)
13  library(stringr)
14  library(XML)
15  library(caret)
16  library(httr)
17  library(SnowballC)
18  # Libreria para realizar actividades en excel: install.packages("xlsx")
19  library(xlsx)
20  library(rJava)
21  library(lattice)
22  # Librerias para realizar Streaming
23  library(ROAuth)
24  library(streamR)
25  # Libreria para importar desde excel haja de datos con data frame
26  library(RJavaTools)
27  library(openxlsx)
```

4.3.3 Autenticación a Twitter

Para lograr conectarse a Twitter en necesario tener cuenta una cuenta, pero además crear una aplicación como desarrollador, ingresando a www.apps.twitter.com. De esta manera obtenemos los valores para las variables que requiere el comando de R setup_twitter_oauth.

Los valores asignados a cada variable han sido reemplazados, los que se publican solo tienen sentido ilustrativo.

```
consumer_key <- 'aaaaaaaaaaaaaaaaaaaaaaaa'
consumer_secret <- 'bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb'
access_token <- '000000000-ccccccccccccccccccccccccccccccccccccccc'
access_secret <- 'ddddddddddddddddddddddddddddddddddddddddddddddddd'

setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)
```

4.3.4 Almacenamiento

A continuación se cargan los términos clave para realizar la consulta a la base de información de

```
#Cargar desde archivo CSV el vector con cadenas para realizar consulta
df_cadenas <- read.csv("c:/DominiosProyectoBigData/cadenas_csv.csv", header=TRUE)
#Toma los registros con las palabras pre seleccionadas del vocabulario objetivo
df_cadenas_seleccionadas <- subset(df_cadenas,seleccionada == "1")
#Toma unicamente el listado de las palabras y las convierte en vector
cadenas <- as.vector(df_cadenas_seleccionadas$cadena)
# Obtener la longitud del vector
# Numero de tweets a obtener
n <- 8000;
# Mensajes a pantalla
sprintf("Consultas a realizar: %d",length(cadenas))
sprintf("Tweets a buscar por consulta: %d",n)
# Crea data frame vacios
dft <- data.frame();
df <- data.frame();
```

```
# Ciclo para extraer tweets y agregarlos al data frame
for (i in 1:length(cadenas))
{
  print("Cadena:");
  print(cadenas[i]);

  tweets <- searchTwitter(cadenas[i], n, lang = 'es')

  print("tweets (registros) encontrados:");
  print(length(tweets));
  print("-------------------------------");
  if (length(tweets)>0)
  {
    df = twListToDF(tweets);
    dft = rbind(dft,df);
  }
}
sprintf("Total de registros encontrados: %d",length(dft$text));

# Se crea un Data Frame de respaldo original
dft_original <- dft;
# Se crea un Data Frame de respaldo
dft_copia <- dft;
# Se crea un archivo CSV con los registros obtenidos
write.csv(dft,"c:/DominiosProyectoBigData/tmp_csv.csv")
```

Twitter, de este lexicón forman parte además de los términos para referirse al Deportivo Pereira, otros más para referirse a otros equipos de la categoría A con incidencia en el área metropolitana de Pereira.

Dentro de esta fase del proceso se obtiene el porcentaje de mensaje duplicados dentro del conjunto obtenido, un mensaje es generado por un usuario y se convierte en un mensaje repetido porque otros usuarios lo han retwitteado.

```
Source on Save | Run | Source
# Se carga el archivo
df_cargado<-read.csv("c:/DominiosProyectoBigData/tmp_csv.csv", header=TRUE)
dft <- df_cargado;
dft_DataSet <- df_cargado;
sprintf("Total de registros cargados: %d",length(dft$text));

# ELIMINACIÓN DE DUPLICADOS
#Codigo para eliminar elementos duplicados de un data frame
sprintf("Total de registros CON registros REPETIDOS: %d",length(dft$text));
dft_no_duplicados <- dft[!duplicated(dft$text), ]
sprintf("Total de registros SIN registros REPETIDOS: %d",length(dft_no_duplicados$text));
sprintf("Total de registros REPETIDOS: %d",length(dft$text)-length(dft_no_duplicados$text));
sprintf("Porcentaje de registros REPETIDOS: %.2f%%",(1-(length(dft_no_duplicados$text)/length(dft$text)))*100);
```

4.3.5 Técnicas de filtrado

Ahora corresponde realizar el filtrado, se eliminan los RT de los mensajes, enlaces, emoticones, signos de puntuación, se pasa el texto a minúsculas, con este filtro se crea el Data Frame semi filtrado, luego se continua eliminando las menciones a otros usuarios y las palabras vacías del idioma español, finalmente se guarda como el Data Frame filtrado. Los verbos no son convertidos en su forma infinitiva por considerarse que el mensaje puede cambiar de sentido.

```r
# Cargar mensajes del DataSet
df_cargado <- dft;
length(df_cargado$text)
# Eliminar/remover RT de los mensajes
#corpus_tweets = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "",dft$text )
dft$text = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "",dft$text)
# Eliminar/remover enlaces
dft$text = gsub("(http://|https://)\\w+", "", dft$text)
# Elimina Emoticones
dft$text <- str_replace_all(dft$text,"[^[:graph:]]", " ")
# Eliminar/remover signos de puntuacion
dft$text <- gsub("[[:punct:]]", "", dft$text)
# Convierte Columna text en minuscula
dft$text <- tolower(dft$text);
# Se guarda el data frame semi filtrado
dft_semi_filtrado <- dft;
# Solo correr este filtro para agrupar las palabras más usadas
# Eliminar/remover menciones a otros usuarios
dft$text = gsub("@\\w+", "", dft$text)
# Solo correr este filtro para agrupar las palabras más usadas
# Elimina stopwords o palabras vacias !Verificar si vale la pena o elimina valiosa informacion!
dft$text <- tm::removeWords(x = dft$text, stopwords("spanish"))
dft_filtrado <- dft;
# El DataSet orginial regresa a dft
dft <- df_cargado;
```

4.3.6 Análisis con R

Se debe cargar el lexicón con el vocabulario del dominio del problema, se encuentra en un archivo CSV.

```r
#Carga vocabulario caracteristico del Dominio investgado de archivo CSV
df_vocabulary <- read.csv("c:/DominiosProyectoBigData/vocabulario_csv.csv", header=TRUE)
#Toma los registros con las palabras pre seleccionadas del vocabulario objetivo
df_listado <- subset(df_vocabulary,Seleccionada == "1")
#Toma unicamente el listado del vocabulario y las convierte en vector
listado <- as.vector(df_listado$word)
df <- dft_semi_filtrado;
```

Una vez cargado el lexicón del vocabulario, se seleccionan los mensajes que contengan dichas palabras y se almacenan en el Data Frame sub_data Luego se crea un archivo tipo CSV con los mensajes que contienen palabras del lexicón.

```
#Expresion aplicando operador OR
sub_data <- subset(df, grepl(paste(listado,collapse = "|"),text))
sprintf("Registros encontrados: %d",length(sub_data$text));
length(sub_data$text);
#Crea archivo CSV con los mensajes que contienen el vocabulario seleccionado
df_subconjunto_dominio <- sub_data;
write.csv(df_subconjunto_dominio,"c:/DominiosProyectoBigData/conocimiento_csv.csv")
```

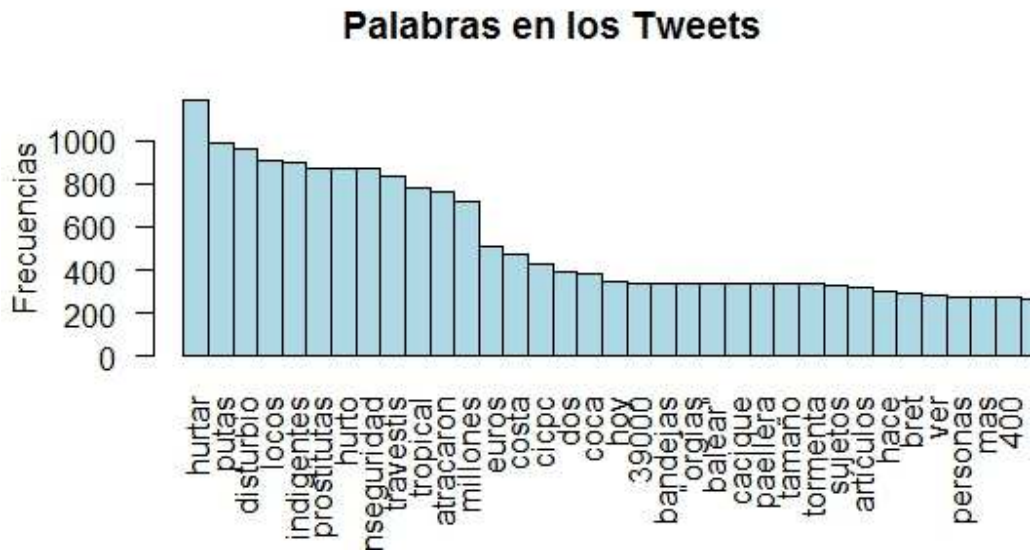
A continuación se deben considerar las palabras más utilizadas dentro del nuevo Data Set guardado como archivo CSV, para que sean incluidas dentro del vocabulario por el analista humano.

```
Source on Save
# Cargar el Data Set
dft <- df_filtrado;
# Crea un corpus con los mensajes de la base de datos obtenida
corpus_tweets = Corpus(VectorSource(dft$text));
# Crea el tdm Matriz de terminos de documento
tdm <- TermDocumentMatrix(corpus_tweets);
# Crea una matriz de un conjunto de valores
matriz = as.matrix(tdm);
# Orden de uso de palabras de forma decreciente
matriz_f <- sort(rowSums(matriz),decreasing=TRUE);
# Crea un data frame con las palabras y sus frecuencias
df_frecuencias <- data.frame(word = names(matriz_f), freq=matriz_f);
# Escribe la tabla de palabras vrs. frecuencias
table(df_frecuencias)
length(df_frecuencias$word)
head(df_frecuencias)
#Crea archivo CSV con las palabras del data set
write.csv(df_frecuencias,"c:/DominiosProyectoBigData/vector_csv.csv")
```

Finalmente se imprime un gráfico con las frecuencias de las palabras.

```
20 # Imprimir grafico de barras con las frecuencias de las palabras
21 barplot(df_frecuencias$freq,space=0,horiz=FALSE,
22         names=df_frecuencias$word,main='Palabras en los Tweets',
23         xlab='',ylab='Frecuencias',col='lightblue',las=2,xlim = c(0,29))
```

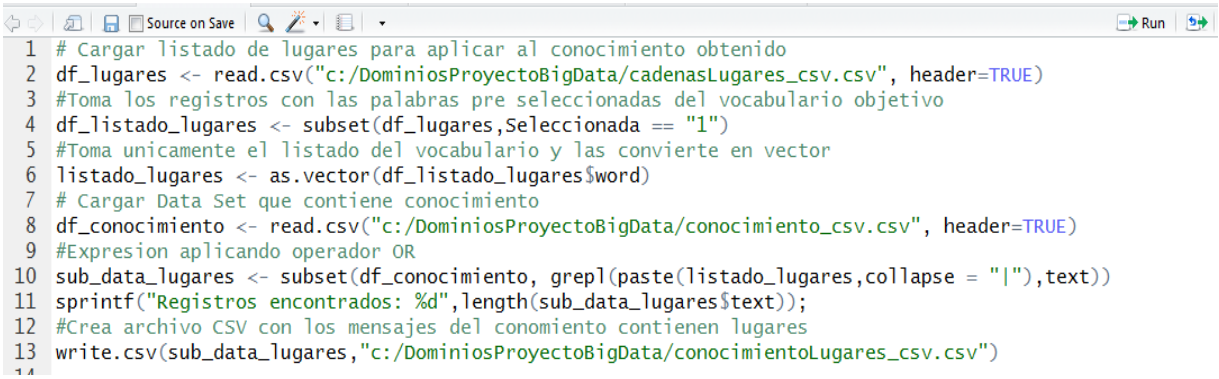
Que se representa en el siguiente ejemplo.



4.3.7 Conocimiento

De la fase anterior se ha obtenido un conocimiento, sin embargo dada la forma de cómo se generaron los mensajes por los usuarios de la red social, no contienen en su mayoría ubicación para geo referenciar algún tipo de actividad, por lo que se ha hecho necesario crear un lexicón con los lugares del área metropolitana de Pereira en los que han ocurrido incidentes que afectaron la convivencia y seguridad ciudadana por situaciones derivadas de las barras bravas del futbol. De esta manera se filtra nuevamente el Data Set para obtener el conocimiento final con

mensajes de un comportamiento que afecta la convivencia y seguridad ciudadana en lugares del área metropolitana expresados explícitamente.



```r
# Cargar listado de lugares para aplicar al conocimiento obtenido
df_lugares <- read.csv("c:/DominiosProyectoBigData/cadenasLugares_csv.csv", header=TRUE)
#Toma los registros con las palabras pre seleccionadas del vocabulario objetivo
df_listado_lugares <- subset(df_lugares,Seleccionada == "1")
#Toma unicamente el listado del vocabulario y las convierte en vector
listado_lugares <- as.vector(df_listado_lugares$word)
# Cargar Data Set que contiene conocimiento
df_conocimiento <- read.csv("c:/DominiosProyectoBigData/conocimiento_csv.csv", header=TRUE)
#Expresion aplicando operador OR
sub_data_lugares <- subset(df_conocimiento, grepl(paste(listado_lugares,collapse = "|"),text))
sprintf("Registros encontrados: %d",length(sub_data_lugares$text));
#Crea archivo CSV con los mensajes del conomiento contienen lugares
write.csv(sub_data_lugares,"c:/DominiosProyectoBigData/conocimientoLugares_csv.csv")
```

Capítulo 5
Análisis de Resultados y Conclusiones

5.1 Recopilación y lectura de la información

La recopilación de la información objeto de ser examinada ha iniciado con un año de anticipación, éste conjunto de datos cuenta con por lo menos 432.700 registros o tweets, tomados de la red social de Microblogging Twitter en distintas fechas; recolección realizada cada vez que un encuentro deportivo del futbol profesional se realizara.

Las consultas se lanzaron aproximadamente entre 12 y 48 horas luego de terminados los encuentros deportivos con el fin de capturar las reacciones y situaciones posteriores al encuentro deportivo, así como anteriores y durante tales encuentros.

Pasadas ciertas observaciones y en vista que las consultas centradas en el equipo Deportivo Pereira arrojaron pocos registros en la red social Twitter, se toma la determinación de ampliar las consultas también a equipos del futbol colombiano tales como América de Cali, Deportivo Cali, Atlético Junior y Atlético Nacional desde el mes de marzo de 2017 hasta el presente mes de Noviembre de 2017. La tabla 3 muestra los criterios de búsqueda seleccionados.

Categorías de búsqueda				
lobo sur pereira	corpereira	america de cali	barón rojo	hincha robando
lobo sur	hincha pereira	américa de cali	deportivo cali	hinchas atracando
deportivo pereira	hinchas pereira	la mechita	atletico junior	hincha atracando
furia matecana	atletico nacional	hincha américa	atlético junior	hinchas robar
lsp	atlético nacional	hincha america	equipo tiburón	hincha robar
super depor	barrabrava nacional	hinchas américa	equipo tiburon	hinchas ladrones
ldsp	hincha nacional	hinchas america	junior de barranquilla	hincha ladron
barra brava pereira	hinchas nacional	baron rojo	hinchas robando	

Tabla 3. Listado de categorías de búsqueda.

Con los criterios de búsqueda de la tabla 3 se han obtenido por categoría el siguiente número de mensajes del Data Set propio de la red social Twitter, se presenta en la tabla 4. El bajo número de mensajes relacionados con el Deportivo Pereira ratifica la importancia de incluir en los criterios de búsqueda a otros equipos del futbol profesional colombiano que cuentan con barras dentro del área metropolitana.

En la consola de resultados se puede observar la respuesta entregada por el sistema.

```
[1] "Cadena:"
[1] "corpereira"
[1] "tweets (registros) encontrados:"
[1] 439
[1] "--------------------------------"
[1] "Cadena:"
[1] "hincha pereira"
[1] "tweets (registros) encontrados:"
[1] 34
[1] "--------------------------------"
[1] "Cadena:"
[1] "hinchas pereira"
[1] "tweets (registros) encontrados:"
[1] 92
[1] "--------------------------------"
[1] "Cadena:"
[1] "atletico nacional"
[1] "tweets (registros) encontrados:"
[1] 8000
```

No.	Cadena	Mensajes	No.	Cadena	Mensajes
1	atletico nacional	8000	21	hinchas robando	76
2	hinchas nacional	8000	22	barrabrava nacional	61
3	america de cali	8000	23	hincha robar	42
4	junior de barranquilla	8000	24	**hincha pereira**	34
5	hincha nacional	5945	25	hincha ladron	24
6	hincha america	5935	26	**furia matecana**	16
7	hinchas america	5432	27	hincha robando	16
8	la mechita	4808	28	hinchas atracando	6
9	deportivo cali	2907	29	**ldsp**	2
10	atletico junior	1659	30	**barrabrava pereira**	1
11	equipo tiburon	1182	31	barón rojo	1
12	baron rojo	774	32	hincha atracando	1
13	**corpereira**	439	33	**lobo sur pereira**	0
14	hinchas robar	419	34	atlético nacional	0
15	**deportivo pereira**	319	35	américa de cali	0
16	**lobo sur**	171	36	hincha américa	0
17	hinchas ladrones	171	37	hinchas américa	0

18	**super depor**	155	38	atlético junior	0
19	**lsp**	98	39	equipo tiburón	0
20	**hinchas pereira**	92	40		

Tabla 4. Muestra de mensajes obtenidos por categoría.

La consulta se realiza con 39 criterios (ver tabla 3), se obtienen 62.786 tweets (ver tabla 4) en 59 minutos y 50 segundos. Para cada criterio se solicitaron 8.000 mensajes y los equipos Nacional, América y Junior los reunieron, el Deportivo Pereira apenas reúne 439 con corpereira.

Una de las consideraciones importantes tenidas en cuenta en el presente proyecto es la ubicación desde la que se crean los mensajes de Twitter, sin embargo ha debido ser replanteado ya que de los 432.700 mensajes obtenidos al momento (recolección realizada durante varios meses), apenas 548 cuentan con información de los campos longitud y latitud, ello corresponde apenas al 0.12% del conjunto de datos, cantidad de mensajes muy baja con esta característica para ser considerada relevante.

Por otro lado, los registros obtenidos consulta tras consulta se almacenaron en una hoja de cálculo, la seleccionada ha sido Microsoft Office Excel 2010. Cada consulta a la base de datos de Twitter se realizó por medio del lenguaje estadístico R utilizando la API TwitteR, tales datos obtenidos se capturan en estructuras de datos denominadas “Data Frame” las cuales son exportadas a la hoja de cálculo en formato .xlsx. Recuperar desde archivo un conjunto de datos de 30.000 registros toma 60 minutos en una computadora portátil convencional con procesador Corei5 de 4 GB en memoria RAM. Sin embargo no ha sido posible recuperar el DataSet de poco más de 116.000 registros, por lo que fue necesario cambiar el formato de almacenamiento para el archivo por CSV, de ésta forma se logra disminuir sustancialmente el tiempo de carga de más de

116.000 registros cada uno con 16 campos, a tan solo 15 segundos (en la computadora descrita anteriormente).

Una vez sorteada la dificultad para leer desde archivo el Data Set y además poderlo cargar en R en menos de un minuto, se procede a determinar el porcentaje de mensajes repetidos. Obteniendo la información listada en la tabla 5 así:

Concepto	Número de mensajes	Valor porcentual
Mensaje repetidos	277.879	64%
Mensajes no repetidos	154.819	36%
Totales	432.698	100%

Tabla 5. Cantidad de mensajes repetidos dentro del Data Set.

A continuación se procede a realizar la etapa de filtrado, en la cual se eliminan o remueven de los mensajes las siguientes características:

- RT (Retweets): Comunicar un mensaje de otro usuario desde la cuenta propia.
- Menciones: Incluir dentro de un mensaje propio a otro(s) usuario(s) de Twitter.
- Signos de puntación
- Enlaces: Direcciones URL que hacen referencia a otros recursos y que hacen parte del mensaje.
- Emoticones: Conjuntos de caracteres que representan gestos por medio de caras.
- Mayúsculas: Se convierte cada mensaje Tweet a letras minúsculas.
- Palabras vacías del castellano: Conocidas como “stop words”, aquellas palabras que carecen de significado por si solas, como artículos, preposiciones, conjunciones, pronombres, entre otros.

La etapa de filtrado entrega una nuevo Data Set, el cual es un arreglo de información de tipo Data Frame al cual se le da el nombre data frame filtrado, o df_filtrado por facilidad, además se genera un Data Frame semi filtrado denominado dft_semi_filtrado el cual contiene elementos valiosos para un lector humano tales como las palabras vacías, enlaces, signos de puntuación y menciones.

Ahora para el análisis de la información, se realiza la búsqueda con palabras claves con las que se puedan establecer relaciones, para ajustar mejor el vocabulario base se realizó la siguiente actividad:

- Se toma el campo “text” correspondiente al mensaje de cada registro, el campo text contiene el mensaje o tweet.
- Se crea una matriz con cada una de las palabras de los mensajes y sus respectivas frecuentas o pesos.
- Finalmente este listado es ordenado en orden decreciente y se seleccionan y marcan las palabras más relevantes que se consideran relevantes dentro del contexto de las barras de futbol y contravenciones que puedan afectar la convivencia y la seguridad ciudadana.

Ésta matriz contiene el listado de palabras del Data Set filtrado con sus respectivas frecuencias o pesos, presentados en orden decreciente. En total son 7.056 diferentes palabras. Aquella con más peso es la palabra “hinchas” con peso 8.003, le sigue la palabra “hincha” con 994. La tabla 6 presenta más de ellas.

No.	Palabra	Frecuencia	% frecuencia
1	hinchas	8.003	2%
2	hincha	994	0.23%
3	partido	692	0.16%
4	vida	554	0.12%
5	robando	423	0.10%
6	disturbios	406	0.09%
7	muerto	387	0.08%
8	barra	385	0.08%
9	camisetas	383	0.08%
10	estragos	363	0.08%
11	violentos	362	0.08%

Tabla 6. Listado de palabras con más frecuencia dentro de data set filtrado.

A continuación en la tabla 7 el listado extendido de las primeras 100 palabras obtenidas.

No.	Palabra	Frecuencia	% Frecuencia	No.	Palabra	Frecuencia	% Frecuencia
1	hinchas	8003	1.850%	51	millonarios	272	0.063%
2	cali	6631	1.532%	52	finales	271	0.063%
3	deportivo	3150	0.728%	53	muere	255	0.059%
4	nacional	2240	0.518%	54	gracias	247	0.057%
5	mechita	2054	0.475%	55	colombiano	242	0.056%
6	pereira	1022	0.236%	56	ciudad	241	0.056%
7	hincha	994	0.230%	57	celebran	239	0.055%
8	equipo	828	0.191%	58	desmanes	238	0.055%
9	america	804	0.186%	59	sanciones	234	0.054%
10	dimayor	797	0.184%	60	ganaron	231	0.053%
11	pascual	790	0.183%	61	peleas	228	0.053%
12	partido	692	0.160%	62	campo	223	0.052%
13	guerrero	599	0.138%	63	robar	221	0.051%
14	sur	585	0.135%	64	sancionar	218	0.050%
15	vida	554	0.128%	65	balacera	216	0.050%
16	mierda	545	0.126%	66	jugadores	212	0.049%
17	colombia	503	0.116%	67	barras	209	0.048%

18	blancas	478	0.110%	68	camiseta	191	0.044%
19	robando	423	0.098%	69	juegos	188	0.043%
20	rojo	417	0.096%	70	culo	182	0.042%
21	disturbios	406	0.094%	71	buga	178	0.041%
22	liga	399	0.092%	72	partidos	175	0.040%
23	jugar	396	0.092%	73	seguidor	169	0.039%
24	muerto	387	0.089%	74	pelea	162	0.037%
25	barra	385	0.089%	75	enfrentamientos	160	0.037%
26	camisetas	383	0.089%	76	pierdan	158	0.037%
27	prensa	382	0.088%	77	descender	155	0.036%
28	verdes	380	0.088%	78	triunfo	153	0.035%
29	bucaramanga	372	0.086%	79	armas	152	0.035%
30	millos	368	0.085%	80	copa	152	0.035%
31	azules	365	0.084%	81	cobraron	150	0.035%
32	estragos	363	0.084%	82	alado	148	0.034%
33	rcn	362	0.084%	83	dim	148	0.034%
34	violentos	362	0.084%	84	delincuentes	147	0.034%
35	lastima	349	0.081%	85	lamentable	142	0.033%
36	juego	330	0.076%	86	puta	141	0.033%
37	falta	326	0.075%	87	metieron	140	0.032%
38	video	322	0.074%	88	penal	138	0.032%
39	mayor	321	0.074%	89	gol	135	0.031%
40	sorprenden	317	0.073%	90	tradicional	135	0.031%
41	cancha	315	0.073%	91	robaron	131	0.030%
42	culos	314	0.073%	92	fuerza	130	0.030%
43	radio	313	0.072%	93	problemas	128	0.030%
44	pasaron	302	0.070%	94	choque	127	0.029%
45	probar	287	0.066%	95	hinchadas	127	0.029%
46	violencia	287	0.066%	96	batalla	126	0.029%
47	estadio	281	0.065%	97	apoyo	124	0.029%
48	equipos	280	0.065%	98	verde	124	0.029%
49	heridos	278	0.064%	99	locura	123	0.028%
50	comportamiento	275	0.064%	100	ratas	123	0.028%

Tabla 7. Listado de las 100 palabras con más frecuencia dentro de Data Set.

Dicha matriz es exportada a disco duro en un archivo en formato CSV. A este nuevo archivo CSV se le crea una nueva columna (campo) el cual es marcado con 1 si se considera que es una

palabra relevante que pueda estar en un mensaje con contenido ofensivo y que de indicios de una contravención que afecte la convivencia y seguridad ciudadana. La tabla 8 presenta ejemplos.

Se hace necesario realizar la lectura de cada una de las 7.056 palabras para marcar en el campo "Seleccionada" si se considera una palabra que pueda hacer parte de mensajes con contenido ofensivo o que pueda hacer parte de un mensaje relacionado con una contravención policial o que afecte la convivencia y seguridad ciudadana.

No.	Palabra	Frecuencia	Seleccionada
1	Hichas	8.003	0
2	Deportivo	1.556	
3	Robando	358	1
4	Violentos	320	1
5	Normal	318	0

Tabla 8. Listado de palabras con frecuencia y marca por contenido ofensivo, dentro de data set filtrado.

De esta manera se consiguen 160 palabras más para el vocabulario, las cuales se cargan (importan) desde archivo a R y se almacenan en un vector, al cual se denomina "vocabulario". Dicho archivo se encuentra en formato CSV y es una colección que cuenta con 790 palabras obtenidas de los mensajes del Data Set, vocabulario de la jerga local, vocabulario escuchado de los barristas y de los términos obtenidos del libro "El lenguaje del hampa y del delito" primera edición, escrito por Manuel Antonio Arias Echeverry; pero de ellas apenas 500 son las tenidas en cuenta para el análisis de información dentro del dominio del problema.

Al analizar el Data Set filtrado con el vocabulario del dominio del proyecto, se obtiene un nuevo Data Set denominado df_subconjunto_dominio con 174.598 mensajes, es decir obteniendo el

40% de mensajes del total del Data Set principal. Sin embargo por su extenso número de mensajes se hace necesario verificar el vocabulario y se desmarcan (eliminan) algunas palabras entre ellas “hincha”, con lo que se obtienen 1.003 mensajes. Dicho Data Set es exportado como archivo CSV con el nombre conocimiento_csv.csv y el cual se presenta al analista humano en orden cronológico para su revisión.

Un reporte adicional denominado conomientoLugares.csv es entregado al analista humano, con apenas 30 mensajes ordenados cronológicamente, contiene aquellos mensajes en los se presume puede ocurrir una afectación a la convivencia y seguridad ciudadana haciendo referencia a un sitio de la ciudad.

El siguiente listado muestra quince de los mensajes obtenidos dentro del conocimiento.

5922	a cada vándalo “hincha que este haciendo daño riñas o robando con la camiseta de x equipo de fútbol se le debe prohibir entrada a estadio
3850	corpereira y asi piden q volvamos al estadio ladrones y falsos que es lo q son estos hpts continúa el círculo vicioso de mierda
3984	un estadio lleno de negros asquerosos y los hijos de puta de corpereira son capaces de hacer una cochinada más grande
4272	corpereira puede entrar el papa y que mierda burros ladrones que se llene el estadio pero pa voliarles monedas hpts
4608	y sigan llendo al estadio a llenarles los bolsillos a esta manada de payasos hijueputas de corpereira
5051	sebaspuerta212 aunque con la recocha que es corpereira ya sabemos que no habrá ascenso por tu carrera busca un… cowjt4zmumbh
5897	abucaramanga andresmarocco hinchas d bucaramanga desde horas d la mañana peleando y robando a estudiantes d colegios alrededor dl es
7	ojo pelea de hinchas en los semaforos de la viña policiapereira
10	hichas del pereira robado por el parque banderas
11	atención hinchas roban en el centro de pereira
13	vuelve la normalidad al centro casi siempre roban hinchas en las reuniones del parque banderas
14	hinchas ampones salen a las calles a robar pilas por el coliseo mayor
15	hinchas del pereira robando en estacion de megabus parque banderas
16	en las calles alerta roja los delincuentes vestidos de hinchas robando reunion hinchas del pereira
17	hinchas del pereira piden limosna y roban alrededores del parque banderas

Es importante destacar que para simular y verificar los análisis del presente proyecto se han agregado de manera controlada diecisiete tweets, siendo algunos de ellos los últimos ocho del anterior reporte. El siguiente es el listado de los mensajes auto creados para la simulación y utilizados como señuelos.

5.2 Conclusiones

El proyecto ha planteado la siguiente hipótesis:

¿Es posible desarrollar un modelo de seguimiento a las barras bravas del futbol basado en tecnologías Big Data, para la ciudad de Pereira?

Si bien dentro del campo de la minería de datos y dentro de la minería de textos se puede investigar para conseguir un modelo de seguimiento a las barras bravas, en el presente proyecto se ha optado por el filtrado y la estadística para lograr responder a la hipótesis. A pesar de contar con registros de 16 campos para el caso de los mensajes de Twitter, es en el campo textual en donde se centra el estudio y es en él en donde emergen una serie de dificultades como las siguientes, las cuales en su momento se abordaron para minimizarse:

- Falta de estructura del texto, luego del filtrado de los mensajes se encuentra perdida de información importante para el contexto y la interpretación.

- El análisis de los textos va a depender del contexto y del dominio del problema, lo cual implica el uso de lexicones o tesauros específicos para el contexto, como los propuestos dentro del presente trabajo.

	text
1	Esto no se queda asi afuera nos vemos putos de nacional
2	vamos por el trapo de nacional semaforos la viña
3	vamos por el trapo de nacional semaforos la viña
4	en la cancha y en la calle todo por mi pereira
5	perros de nacional chuzo les va a sobrar
6	en la juega con los tombos que estan afuera
7	Ojo pelea de hinchas en los semaforos de la viña. @PoliciaPereira
8	Que inseguridad, pelea de hinchas pereira y nacional
9	Una fiesta deportiva se traduce en violencia de hichas pereira y nacional
10	Hichas del Pereira robado por el parque banderas
11	Atención, hinchas roban en el centro de Pereira
12	Falsos hinchas del Pereira robando en el centro
13	Vuelve la normalidad al centro, casi siempre roban hinchas en las reuniones del parque banderas
14	Hinchas ampones salen a las calles a robar, pilas por el coliseo mayor
15	Hinchas del pereira robando en estacion de megabus parque banderas
16	En las calles alerta roja. Los delincuentes vestidos de hinchas robando, reunion hinchas del pereira
17	Hinchas del Pereira piden limosna y roban alrededores del parque banderas

A pesar que el Deportivo Pereira no hace parte de los equipos de la categoría profesional del futbol colombiano, algunas barras de los equipos profesionales muestran su dinámica e influencia en el área metropolitana, por lo que se considera necesario observar su comportamiento dentro del desarrollo del proyecto.

Los mensajes en Twitter tomados para ser observados en su mayoría no contenían información de geo referenciación, apenas el 0.12% de ellos la incluía, por lo que se hace necesario tener una base de conocimiento o lexicón de algunos lugares de la ciudad de Pereira en los que con anterioridad se presentan confrontaciones entre hinchas, así como una forma genérica para identificarlos. Entre estos lugares se tienen los mostrados en la tabla 9, así:

No.	Lugar
1	megabus
2	megabús
3	mega bús
4	mega bus
5	mega
6	viaducto
7	estadio
8	hernan ramirez
9	ciudad victoria
10	la popa
11	el victoria
12	av el pollo
13	avenida el pollo
14	nacederos
15	avenida
16	popa
17	articulado
18	romelia
19	dosquebradas pereira
20	carrera

21	calle
22	kra
23	la viña
24	semaforo
25	parque
26	centro de pereira
27	la villa
28	gamma

Tabla 9. Listado de lugares en que ocurren confrontaciones.

Para lograr mejor entendimiento del conjunto de mensajes entregado por el estudio (conocimiento) al analista humano, se almacena un Data Set paralelo sin que se le eliminen palabras vacías ni signos de puntuación, con lo que los mensajes pueden permanecer originales y conservar su sentido original, dejando de ser un listado de palabras con un significado que puede ser ambiguo.

La clasificación que realiza el analista humano de las palabras más utilizadas en los mensajes debe ser revisado constantemente, es el caso de la palabra “hincha” que por ser la palabra con más peso dentro del Data Set se encuentra en mensajes que no deben ser clasificados, con lo que tenemos mensajes tipo “falsos positivos”.

El análisis del Data Set realiza búsqueda de lo particular y no de lo general, ya que un hecho que puede afectar la convivencia y seguridad ciudadana no es una tendencia dentro de la red social, todo lo contario ocurre con otro tipo de situaciones como el desacuerdo con las directivas del equipo de futbol, anunciar un gol o en otros contextos el gusto o aceptación de un determinado político. Por ello la importancia de entregar el conocimiento al analista humano con el contenido original del mensaje y con la cantidad de mensajes original, es decir sin eliminar los mensajes repetidos.

Como trabajo futuro se recomienda realizar una comparación entre una ontología y el modelo estadístico para elegir entre ellos el modelo de comportamiento que más se aproxime al comportamiento de las barras de futbol en la ciudad de Pereira. También por medio minería de texto y el modelo de Descubrimiento de Conocimiento en Textos (KDT, *Knowledge Discovery in Text*), el cual es derivado del modelo de Descubrimiento de Conocimiento en Datos (KDD, *Knowledge Discovery in Data*) para descubrir asociaciones desconocidas entre hechos desconocidos. En ambas propuestas para trabajos futuros es importante partir de los lexicones conseguidos dentro del presente proyecto, ya que proporcionan información del dominio del problema.

Durante el desarrollo del presente proyecto se realizó una ponencia para el *"IV Encuentro Nacional de Sistemas y Telecomunicaciones"* realizado en la ciudad de Pereira por la Universidad Católica de Pereira, presentándose el 2 de Septiembre de 2016.

Capítulo 6
Lista de Referencias

6.1 Referencias

[1] Diario La Tarde. (S.F). Caso de violencia que afectó instalaciones de Mega Bus. La Tarde. Recuperado de http://www.latarde.com/multimedia/videos/video-146341-asi-fueron-los-danos-de-hinchas-al-megabus-video

[2] Cuero R., (2013), “La orfandad de la nueva generación”, Bogotá Colombia, Intermedio, pp. 5-7, 48 - 49

[3] El Nuevo Siglo. (21 de octubre de 2013). Pereira es referente en manejo de la seguridad ciudadana. El Nuevo Siglo. Recuperado de http://www.elnuevosiglo.com.co/articulos/10-2013-pereira-es-referente-en-manejo-de-la-seguridad-ciudadana-v%C3%A1squez.html

[4] Diario La Tarde. (S.F). La Tarde. Recuperado de: http://www.latarde.com/noticias/judicial/102759-nuevo-subcomandante-de-la-policia-metropolitana

[5] Castaño, G., Uribe, N.y Restrepo, S., (2014),”Barras bravas en el futbol consumo de drogas y violencia”, Medellín Colombia, Fundación Universitaria Luis Amigó.

[6] Diario La Tarde. (S.F). Hinchas del Pereira habrían asesinado a joven en Valledupar. La Tarde. Recuperado de http://www.latarde.com/noticias/judicial/148570-hinchas-del-pereira-habrian-asesinado-a-joven-en-valledupar

[7] Caracol Radio. (2 de diciembre de 2013). Habitantes de Bosa denuncian constantes riñas entre barras bravas. Caracol Radio. Recuperado de http://www.caracol.com.co/noticias/bogota/habitantes-de-bosa-denuncian-constantes-rinas-entre-barras-bravas/20131202/nota/2028651.aspx

[8] El diario.(9 de julio de 2012). Nacional entrenó el sábado en el Monumental. Colombia. El diario. Recuperado de http://www.eldiario.com.co/seccion/DEPORTES/nacional-entren-el-s-bado-en-el-monumental120709.html

[9] Joyanes, L. (2013). “Big Data, análisis de grandes volúmenes de datos en organizaciones”, Mexico D.F. Mexico, Alfaomega, p. 7.

[10] Wirth, R. y Hipp, J., (2000). “CRISP-DM: Towards a Standard Process Model for Data Mining”, Daimler Chrysler.

[11] Cañón, L. y García, B., (2007). “Estudio de Caso sobre el Fenómeno de Barras Bravas: una Mirada desde la Escuela”, Típica: Boletín Electrónico de Salud Escolar, Volumen 3, Número 2.

[12] Consultora IDC. (20 de noviembre de 2012). “Nuevas tendencias tecnológicas han entrado a AL”. URL: http://mx.idclatin.com/releases/news.aspx?id=1433

[13] Consultora Gartner. (31 de marzo de 2011).“CEO Advisory: “Big Data” Equals Big Opportunity”. URL: https://www.gartner.com/doc/1614215/ceo-advisory-big-data-equals

[14] Consultora McKinsey. (1 de marzo de 2011). “Big Data: The next frontier for innovation, competition, and productivity”. URL: http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation.

[15] Han, H, Yonggang W. y Tat-Seng Chua, X. L., (2014). “Toward Scalable Systems for Big Data Analytics: A Technology Tutorial”. IEEE Digital Library.

[16] Michalik, P., Stofa, J. y Zolotova, I., (2014). “Concept Definition for Big Data Architecture in the Education System”. IEEE Digital Library.

[17] Maldonado, S., (2012), “Analítica web: medir para triunfar”, Madrid España, ESIC.

[18] Lovett, J., (2012), “Social Media, métricas y análisis”, Madrid España, Anaya.

[19] Ghemawat, S., Gobioff H. y Leung S., (2003). “The Google File System”, IEEE.

[20] Dean, J. y Ghemawat, S., (2004). “Map Reduce: Simplified Data Processing on Large Clusters”, IEEE.

[21] Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A. y Gruber, R. (2006). “Bigtable: A Distributed Storage System for Structured Data”, IEEE.

[22] Marisol Collazos. (2006). Sawzall lenguaje de programación nativo de Google. Colombia. Actualidad informática. Recuperado de http://www.marisolcollazos.es/noticias-informatica/?p=19

[23] Twitter. En Wikipedia. Recuperado1 de Julio de 2016. https://es.wikipedia.org/wiki/Twitter

[24] CNN. (10 de febrero de 2016). Desastre en Twitter: pierde millones de usuarios y sus acciones se hunden. Colombia. CNN. Recuperado de http://cnnespanol.cnn.com/2016/02/10/desastre-en-twitter-pierde-millones-de-usuarios-y-sus-acciones-se-hunden/#0

[25]. Boyd Danah (2009). Twitter: “pointless babble” or peripheral awareness + social grooming?. Zephoria. Recuperado de http://www.zephoria.org/thoughts/archives/2009/08/16/twitter_pointle.html